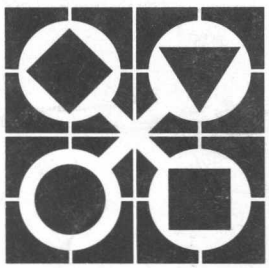
intel

OpenNET™



Data Sheets

TABLE OF CONTENTS

*XENIX is a trademark of Microsoft Corporation.

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and may only be used to identify Intel Products:

BITBUS, COMMputer, CREDIT, Data Pipeline, GENIUS, i, $\hat{\text{i}}$, ICE, iCS, iDBP, iDIS, i^2ICE, iLBX, i_m, iMDDX, iMMX, Insite, Intel, int_el, int_elBOS, Intelevision, int_eligent Identifier, int_eligent Programming, Intellec, Intellink, iOSP, iPDS, iRMX, iSBC, iSBX, iSDM, iSXM, KEPROM, Library Manager, MCS, Megachassis, MICROMAINFRAME, MULTIBUS, MULTICHANNEL, MULTIMODULE, OpenNET, Plug-A-Bubble, PROMPT, Promware, QUEST, QueX, Ripplemode, RMX/80, RUPI, Seamless, SLD, and UPI, and the combination of ICE, iCS, iRMX, iSBC, iSBX, MCS, or UPI and a numerical suffix.

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

* MULTIBUS is a patented Intel bus.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Distribution
Mail Stop SC6-714
3065 Bowers Avenue
Santa Clara, CA 95051

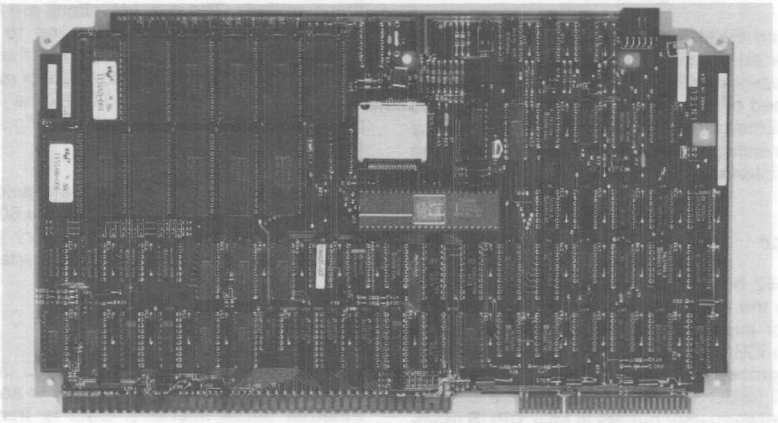
iSBC® 552 AND iSXM™ 552 ETHERNET COMMUNICATIONS ENGINE PRODUCTS

MEMBERS OF THE OpenNET™ PRODUCT FAMILY

- **Provides networking capability for all MULTIBUS® systems regardless of the operating system of the host**
- **Supports XENIX*- and RMX-Network File Service (XNX-NET and RMX-NET) products**
- **Available in two versions**
  - **— Turn-key controller implementing ISO 8073 Class 4 Standard Transport functionality (iSXM™ 552 board) on IEEE 802.3 LANs**
  - **— Flexible, intelligent communications controller for iSBC® 552 board for custom configurations on IEEE 802.3 LANs**
- **iSXM™ 552 board is fully qualified as system extension module for the 86/310, 286/310, 86/380 and 286/380 Intel systems**
- **Resident network software down loaded (SXM) or can be stored in on-board PROMs (SBC)**
- **Runs iNA 960 and iNA 961 (SXM) transport software**
- **On-board diagnostic and boot firmware (SXM)**

The iSBC 552 and iSXM 552 COMMengine products are designed for communications front end processor applications connecting MULTIBUS systems onto IEEE 802.3/Ethernet LANs. COMMengines are dedicated to the communications tasks within a system allowing the host to spend more time processing user applications. A major advantage of COMMengines is that they can be used to network existing systems and established designs without forcing the redesign of the entire system architecture.

The iSBC and iSXM 552 boards can be used with any operating system because they require only a high level interface to communicate with the host (eg. transport commands in case of the iSXM 552 board). The result is a powerful system building block which enables the OEM to connect MULTIBUS-based systems with different operating systems to the same network. Applications for the 552 products include networked multiuser XENIX 286 based systems for the office and iRMX-based systems for real time applications. The iSXM version is a transport engine complete with on board RAM and ROM memory preconfigured to run iNA 961 transport software on-board. iNA 961 software is a version of Intel's iNA 960 LAN software implementing the ISO 8073 Class 4 protocol specially configured to support the iSXM 552 board. The iSBC 552 board is a "de-bundled" version of the iSXM 552 board; it comes without memory and software allowing greater flexibility for the user to adapt the board for his special requirements.



 *XENIX is a trademark of MICROSOFT CORP.

The iSBC® Board vs. the iSXM™ 552 Board

The fundamental difference between the two versions is the iSBC 552 board offers the hardware necessary for the user to construct an Ethernet front-end processor for his unique requirements and the iSXM 552 board provides full ISO standard transport services ready to plug in and to be used without any additional configuration effort. The SXM version is arrived at by populating the iSBC 552 board with 16K bytes of ROMs and 80K bytes of iRAMs, and by providing iNA 961, a directly downloadable transport software module. The iSXM 552 board is configured for Intel's 86/286-310 systems and fully qualified to run in these systems. iSXM 552 customers receive the iNA software with the purchase of the iNA 961 license which is an integral part of the SXM offering.

ARCHITECTURE DESCRIPTION

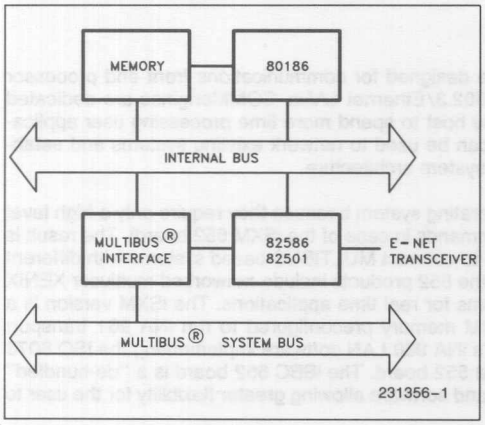


Figure 1.

The iSBC and iSXM 552 boards consist of the following major architectural blocks (see Figure 1): an 80186 processor running at 6 MHz, the Ethernet I/O channel based on the 82586 LAN coprocessor and the 82501 Ethernet serial interface, the on-board memory consisting of ROMs and iRAMs, and the MULTIBUS interface.

Processor

The iSBC 552 board contains an 80186 processor operating in the maximum mode at 6 MHz. It is responsible for implementing the intelligent interface between the iSBC 552 board and a host processor. The 80186 processor runs the iNA 961 (iSXM 552) and iNA 960 (iSBC 552) transport software and delivers data between user buffers in MULTIBUS memory and iNA960/961 buffers on the iSBC and iSXM 552 boards. iNA 960 and 961 software is responsible for the reliable transfer of information across Ethernet.

The 80186 and 82586 both use asychronous ready logic. The 80186 chip select lines are used to select memory mapped I/O locations.

The 80186 supplies the timers and the interrupt controller on iSBC 552. The interrupt controller is used in the fully nested mode. The inputs and the outputs of the 80186 timers are not connected to external sources and destinations. Timer clocking and timer interrupts are generated internally in the 80186.

Memory

The one megabyte address space of the 80186 is divided into four quadrants (see Figure 2). The first (0–256K Byte) and the last (768–1000K Byte) quadrants are reserved for local memory. The second quadrant (256–512K Byte) is used for memory mapped I/O. The iSBC 552 board is totally memory mapped. The third quadrant (512–768K Byte) maps into a 256K Byte MULTIBUS window. This window allows the iSBC 552 board to access a total of 16M Byte of MULTIBUS memory in 256K Byte segments. The iSBC 552 board does not contain any memory which is accessible from MULTIBUS.

The 256K Byte MULTIBUS window starts on 64K Byte boundaries anywhere in the 16M byte MULTIBUS memory. The starting location of this window is determined by a memory mapped I/O latch described in "iSBC 552 User Interface" section.

Local memory on the iSBC 552 board (quadrants one and four) is made up of twelve 28-pin memory sockets. Either EPROM (2764, 27128), Intel iRAM (2186) or equivalent static RAM memory can occupy these sockets. The only limitations are that the lowest pair of sockets corresponding to the bottom memory location must be RAM and the highest pair of sockets corresponding to the top memory location must be EPROM or ROM. The intermediate pairs of sockets can be jumper-configured to be either RAM or EPROM.

Memory mapped I/O locations are selected by the PCS and the MCS control lines of the 80186 processor. Functions controlled by memory mapped I/O are discussed in "iSBC 552 User Interface" section.

Ethernet Interface

The Ethernet Interface on the iSBC 552 is implemented by the 82586 LAN controller and the 82501 Ethernet Serial Interface. Data is transferred be-

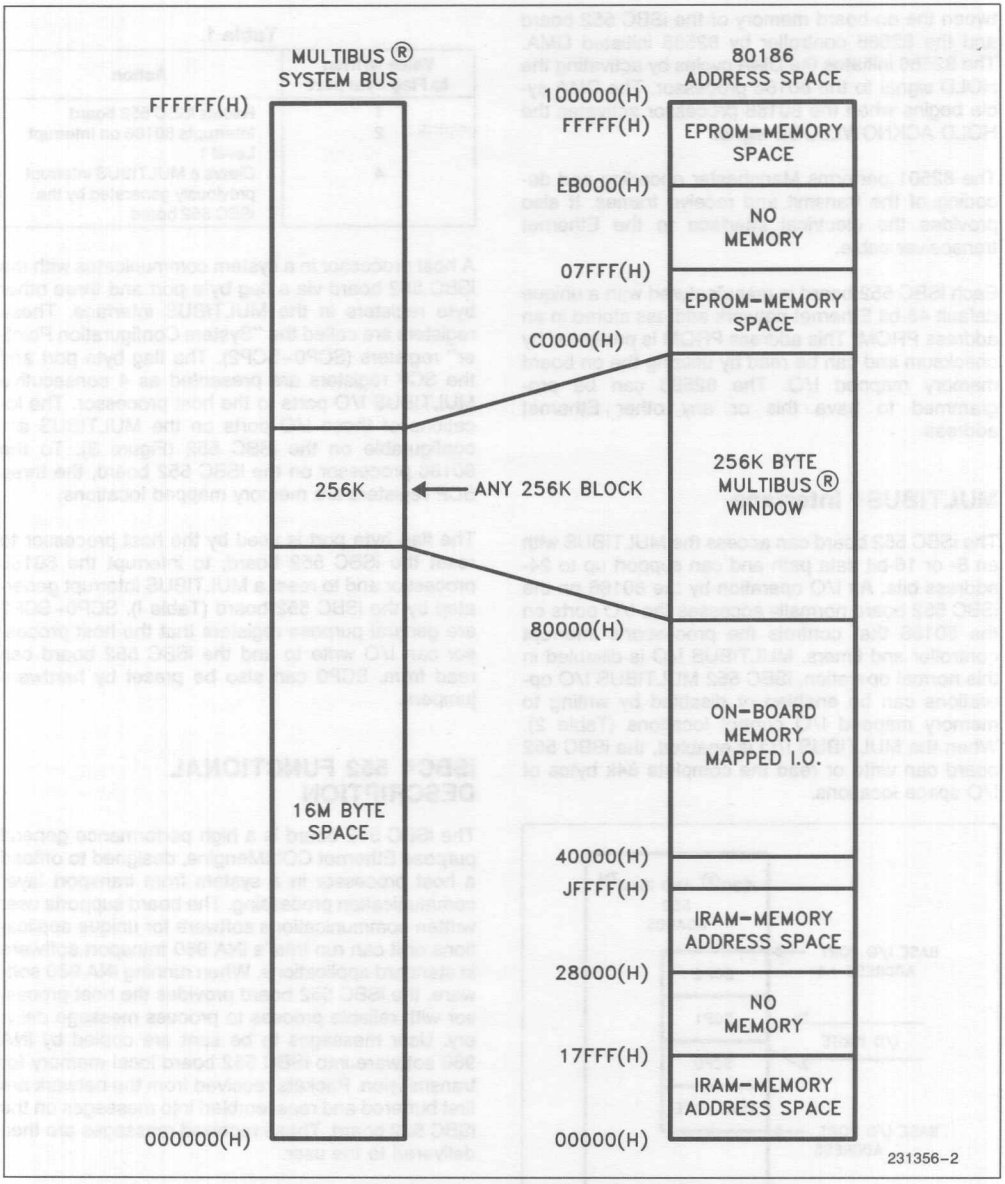


Figure 2. iSBC® 552 Memory Configuration

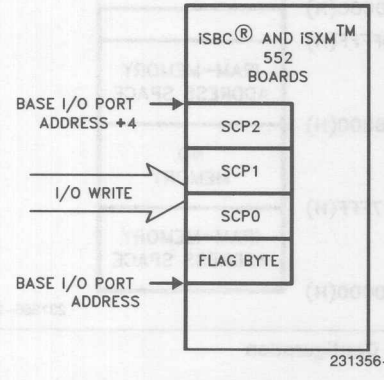
tween the on-board memory of the iSBC 552 board and the 82586 controller by 82586 initiated DMA. The 82586 initiates the DMA cycles by activating the HOLD signal to the 80186 processor. The DMA cycle begins when the 80186 processor activates the HOLD ACKNOWLEDGE signal.

The 82501 performs Manchester encoding and decoding of the transmit and receive frames. It also provides the electrical interface to the Ethernet transceiver cable.

Each iSBC 552 board is manufactured with a unique default 48-bit Ethernet network address stored in an address PROM. This address PROM is protected by checksum and can be read by utilizing the on board memory mapped I/O. The 82586 can be programmed to have this or any other Ethernet address.

MULTIBUS® Interface

The iSBC 552 board can access the MULTIBUS with an 8- or 16-bit data path and can support up to 24-address bits. An I/O operation by the 80186 on the iSBC 552 board normally accesses the I/O ports on the 80186 that controls the processor's interrupt controller and timers. MULTIBUS I/O is disabled in this normal operation. iSBC 552 MULTIBUS I/O operations can be enabled or disabled by writing to memory mapped I/O control locations (Table 2). When the MULTIBUS I/O is enabled, the iSBC 552 board can write or read the complete 64k bytes of I/O space locations.



Base I/O port address = configurable. If 8 bit I/O is used, base port address is configurable from 0-OFCH. If 16 bit I/O is used, base port address is configurable from 0-OFFFCH.
Flag byte: see Table I.
SCP0–SCP2: I/O written by host processor and read by 80186 on iSBC and iSXM 552 SCP0 can be jumper preset.

Figure 3. iSBC® 552 MULTIBUS® Communication Interface

Table 1.

Value written to Flag byte port	Action
1	Resets iSBC 552 board
2	Interrupts 80186 on Interrupt Level 1
4	Clears a MULTIBUS interrupt previously generated by the iSBC 552 board

A host processor in a system communicates with the iSBC 552 board via a flag byte port and three other byte registers in the MULTIBUS interface. These registers are called the "System Configuration Pointer" registers (SCP0–SCP2). The flag byte port and the SCP registers are presented as 4 consecutive MULTIBUS I/O ports to the host processor. The locations of these I/O ports on the MULTIBUS are configurable on the iSBC 552 (Figure 3). To the 80186 processor on the iSBC 552 board, the three SCP registers are memory mapped locations.

The flag byte port is used by the host processor to reset the iSBC 552 board, to interrupt the 80186 processor and to reset a MULTIBUS interrupt generated by the iSBC 552 board (Table I). SCP0–SCP2 are general purpose registers that the host processor can I/O write to and the iSBC 552 board can read from. SCP0 can also be preset by hardware jumpers.

iSBC® 552 FUNCTIONAL DESCRIPTION

The iSBC 552 board is a high performance general purpose Ethernet COMMengine, designed to offload a host processor in a system from transport layer communication processing. The board supports user written communications software for unique applications or it can run Intel's iNA 960 transport software in standard applications. When running iNA 960 software, the iSBC 552 board provides the host processor with reliable process to process message delivery. User messages to be sent are copied by iNA 960 software into iSBC 552 board local memory for transmission. Packets received from the network are first buffered and reassembled into messages on the iSBC 552 board. These received messages are then delivered to the user.

The iSBC 552 board makes use of the functions on the 82586 and 82501 Ethernet controllers to implement a number of network functions. These functions include reprogramming the iSBC 552 station address, Multicast packet reception filtering, Time Domain Reflectometer tests and Loopback diagnostics. The 82586 also records a number of network statistics information. Information stored include the number of CRC and alignment errors, the number of

occurrences of no receive buffer resources and the number of DMA overruns/underruns.

The iSBC 552 can be configured to have a range of local memory configurations, from 16K Byte RAM (160K Byte EPROM/ROM) to 80K Byte RAM (16K Byte EPROM/ROM).

The iSBC 552 board and iNA 960 software combination offers a flexible and configurable transport COMMengine, and allows a user to optimally configure his system for highest performance. The iSXM 552 and iNA 961 combination offers a preconfigured turn-key solution. In both cases, iNA 960 software and the 552 significantly reduces the design cycle involved in designing and implementing a transport COMMengine.

iSBC® 552 User Interface

The iSBC 552 board communicates with a host processor through a handshake of interrupts. The host processor can generate flag byte interrupts to the 80186 on the iSBC 552 and the iSBC 552 can generate MULTIBUS interrupts to the host processor. The host processor and the iSBC 552 can also communicate through shared MULTIBUS system memory. None of the on-board buffer on the iSBC 552 is accessible to the host processor but the iSBC 552 can read and write all of 16M byte of MULTIBUS system memory.

The host processor and the iSBC 552 board further communicate through the SCP registers. These byte registers can be I/O written by the host and can be read through memory mapped I/O by the iSBC 552 processor.

The 80186 processor controls the iSBC 552 through memory mapped I/O. Functions that are controlled are listed in Table 2.

OPERATING ENVIRONMENTS

The iSBC 552 is designed to function in any MULTIBUS systems as a communications processor. It can function as both a MULTIBUS bus master or a slave. As a MULTIBUS master, it can access up to 16M byte of host memory and 64K byte of I/O address. As a MULTIBUS slave, it occupies four consecutive I/O locations on the MULTIBUS. These locations are reserved for the flag byte and the three SCP registers.

iSXM™ 552 FUNCTIONAL DESCRIPTION

The iSXM 552 board is a preconfigured iSBC 552 with 16K Bytes of boot firmware and 80K Bytes of iRAM. The iSXM 552 board is offered with iNA 961 preconfigured ISO 8073 transport software. The iSXM 552 firmware provides the capabilities to load iNA 961 onto the 552 from either a buffer in the local host or remotely from another Ethernet station. It also performs a variety of Ethernet and on-board diagnostics (see sections on iNA 961 User Interfaces and Operating Systems Environment).

iNA 961 software and the iSXM 552 board together provide the functionality of a preconfigured operating system independent transport engine. In addition to transport services, iNA 961 software also includes extensive Data Link and Network Management Facility services. Figure 4 shows the configuration of iNA 961. Table 3 shows some examples of functions provided by iNA 961. iNA 961 is a preconfigured version of iNA 960. Refer to the iNA 960 data sheet for more iNA 961 information.

User programs that use iNA 960 and the iSBC 186/51 board can be run on a host processor with iNA 961 and iSXM 552 as a transport engine. The user programs will require minimal changes in most cases.

Table 2. iSBC® 552 Memory Mapped Functions

80186 Chip Select Lines	Read/Write by 80186	Functions
MCS	R	MULTIBUS® Interface registers (System Configuration Pointer registers, see section 3.4)
PCS	W	Channel Attention to 82586
	R	Reading iSBC® 552 Ethernet Address PROMS
	W	Controlling loopback of 82501
	W	Disabling and Enabling MULTIBUS® I/O
	W	Generating and Clearing iSBC® 552 interrupts to the MULTIBUS® System Bus
	W	Controlling the on-board LED
	W	Latches the MULTIBUS® window segment (8 most significant bits of 24 bit address)

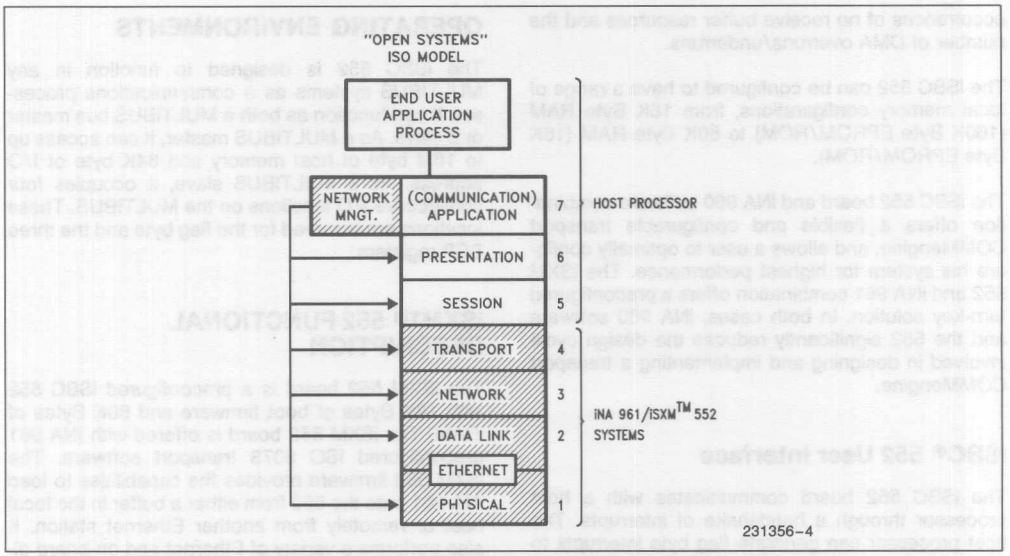


Figure 4. iNA 961 CONFIGURATION ON iSXM™ 552 Board

Table 3. iNA 961 Services

Transport	Virtual circuit open: establish a virtual circuit database send connect: actively try to establish a virtual connection await connect: passively awaits the arrival of a connection request send: send a message receive: post a buffer to receive a message close: close a virtual circuit Datagram send: send a datagram message receive: post a buffer to receive a datagram message
Data Link	Transmit: transmit a data link packet Receive: post a buffer to receive a data link packet Connect: make a data link logical connection (link service access point, IEEE802.3/802.2) Disconnect: disconnect a data link logical connection Change Ethernet address: change the Ethernet address Add multicast address: add a multicast address Delete multicast address: remove a multicast address Configure 82586: configure the 82586 controller
Network Management	Read/Clear/Set network objects (local/remote): read/clear/set local or remote iNA 960 network parameters Read/Set network memory (local/remote): read/set memory of the local or a remote station. Useful in network debug process. Boot consumer: requests a network boot server to load a boot file into this station Echo: Echo a packet between this station and another remote station on the network

iSXM™ Boot Firmware User Interface

The iSXM 552 boot firmware is used to load iNA 961 or other software onto the 552 from either local MULTIBUS memory or a remote network station. The firmware performs a number of local and network diagnostics. Table 4 describes the functions of the boot firmware.

The iSXM 552 boot firmware interfaces with the host processor through a configurable command buffer location in MULTIBUS memory. This location can be either jumper or program configured. The host processor updates the command byte in the command buffer and expects the firmware to update the response byte when the command is done. The host processor signals to the firmware to examine this command buffer by writing a 2 to the flag byte port. The firmware will update the response byte when the command is completed.

The iSXM 552 boot firmware commands fully support the initialization of the MIP Interface. The MIP interface is used by the host processor to communicate with the iNA 961 once it is loaded and started. (See section "iNA 961 User Interfaces for" details.)

iNA 961 User Interfaces

User programs give iNA 960 commands to the iNA 961 software on the iSXM 552 board via the MULTIBUS Interface Protocol (MIP). MIP is an Intel reliable process to process message delivery protocol between MULTIBUS processors. Figure 5 illustrates how this message delivery functions. Commands are passed between the iSXM 552 and the host processor in the form of request blocks. A request block is a buffer that contains a command specification and the command parameters. Each request block (or equivalently, each command) is reliably delivered from the host processor to iNA 961 via the MIP facility. iNA 961 will extract the command information and carry out the command. After a command is done, iNA 961 will use the MIP facility to return the command result to the user program.

iNA 961 request blocks are in the same formats as iNA 960 commands. Refer to the iNA 960 data sheet and reference manuals for more details on iNA 961 software.

Table 4. iSXM™ 552 Boot Firmware Commands

Command	Function
Presence	This command will indicate that the boot firmware is functional by returning the version number of the firmware, the power on diagnostic result, and the default Ethernet address of the iSXM 552.
Load	Load a program from MULTIBUS memory into a designated location in the iSBC 552 memory.
Start	Load a program from MULTIBUS bus memory into a designated location in the iSXM 552 memory. Proceed to start this program once it is loaded. This command also initialize the MIP interface on the iSXM 552 board.
Echo	Echo a packet between this iSXM 552 board and another station on the network.
Remote Boot	This command requests a remote boot server station to download software onto the iSXM 552.
and start MIP initialize	Used after a remote boot. This command initializes the MIP interface on the iSXM552 and then start the software loaded by the remote boot command.

Operating Systems Environment

The iSXM 552 board and iNA 961 software can function in any MULTIBUS environment. The communication between the iSXM 552 and the host processor is entirely independent of any host operating systems. iNA 961 uses the MIP protocol to interface with the host processor. The MIP is a reliable, host operating system independent, process to process communication scheme between any processors on the MULTIBUS System Bus. iNA 961 can service multiple processes utilizing its services at the same time.

A host processor passes iNA 961 commands and buffers in the MULTIBUS system memory to the iNA 961 software. iNA 961 is responsible for updating the response fields of these commands. It is responsible for copying the user send buffer in MULTIBUS system memory into its on board buffers for transmission and for copying received messages to user buffers in MULTIBUS system memory.

Diagnostics

The iSXM 552 board offers a range of power up diagnostics designed to ensure that the 80186 processor, the memory (EPROM and iRAM), and the Ethernet serial interface are functioning properly. Table 5 describes these diagnostics.

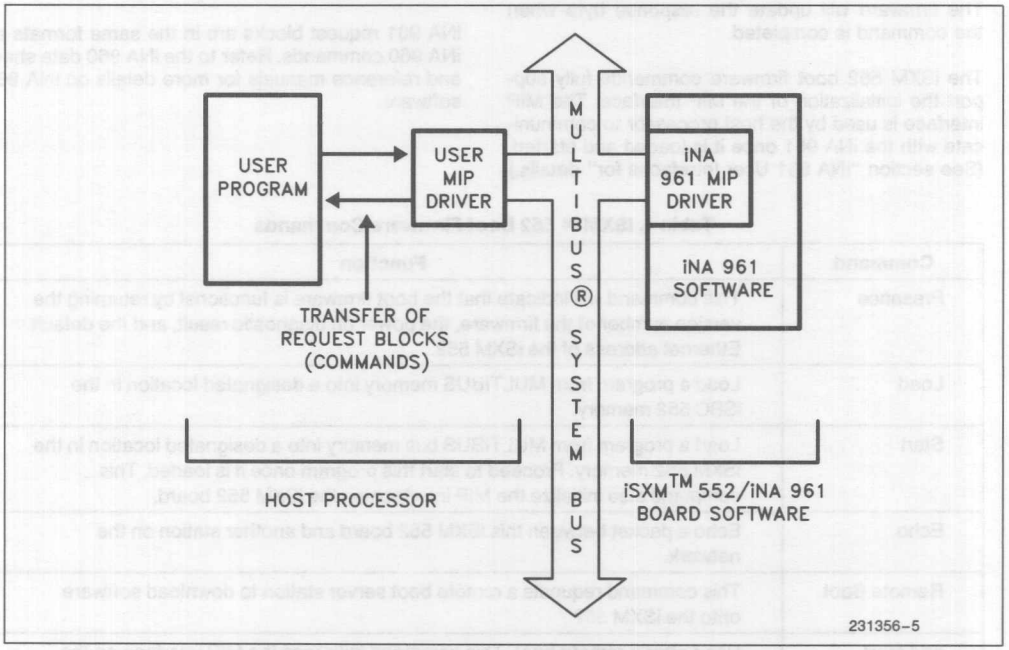


Figure 5. iNA 961 MIP Interface

Table 5. Functions Checked by iSXM™ 552 Diagnostics

1. Insufficient RAM
2. Ram match pattern test
3. Ram ripple data test
4. Boot firmware PROM checksum
5. Address PROM checksum
6. 80186 interrupt controller
7. 80186 timer controller
8. 82586 initialization
9. 82586 CRC check
10. 82586 broadcast packet recognition
11. 82586 external loopback
12. 82586 individual address recognition
13. 82586 multicast address recognition
14. 82586 reset
15. 82586 diagnose check

DEVELOPMENT ENVIRONMENT

The iSXM 552 board is a turn-key product that allows a user to emphasize the development of high level software, such as a network file server. The iSXM 552 board and iNA 961 software together form a transport COMMengine that integrates into any MULTIBUS system. iNA 961 is supplied in a boot loadable file format. This file can be loaded into the iSXM 552 by a host processor or through a remote boot server. The boot firmware on the iSXM 552 supports both functions.

The iSBC 552 allows a user to fine tune iNA 960 and put the software on the board. Both iNA 960 and the iSBC 552 can be flexibly configured to best meet the users' requirements. An Intel development system, together with an Intel I2ICE or equivalent product is usually needed if the user desires to do extensive development work on the iSBC 552. Intel also supplies a wide range of host processor boards and systems (such as the iSBC 286/10 and system 310) that will function well both with the iSBC 552 or the iSXM 552.

iNA 960 can be put into PROMs and run on the iSBC 552.

ORDERING INFORMATION

Part Number	Description
SXM 552	Ethernet Transport Engine
SBC 552	Ethernet COMMengine

SPECIFICATIONS

Data Transfer	Eight or 16-bits.
Average Raw MULTIBUS Transfer Rate	8.7M bits/second (450 ns, 16 bit system memory and no MULTIBUS contention)

Transceiver Interface

Transmit Data Rate	10M bits/second
Signal Levels	Series 10,000 ECL-compatible.
Host Interrupts	One MULTIBUS non-vector interrupt for use in system/host handshaking.
MULTIBUS Interface	The iSBC 552 board conforms to all AC and DC requirements outlined in Intel MULTIBUS Specification, Order Number 142686-002m except for the following signals:

Signal	iSBC Board	MULTIBUS Spec
DATO*– DAT7*	IIH = 180 μA	IIH = 125 μA

DC Power Required	All voltages supplied by the MULTIBUS interface. +5.0V ± 5%, 5.9A maximum +12.0V ± 5%, 0.5A maximum

Environmental

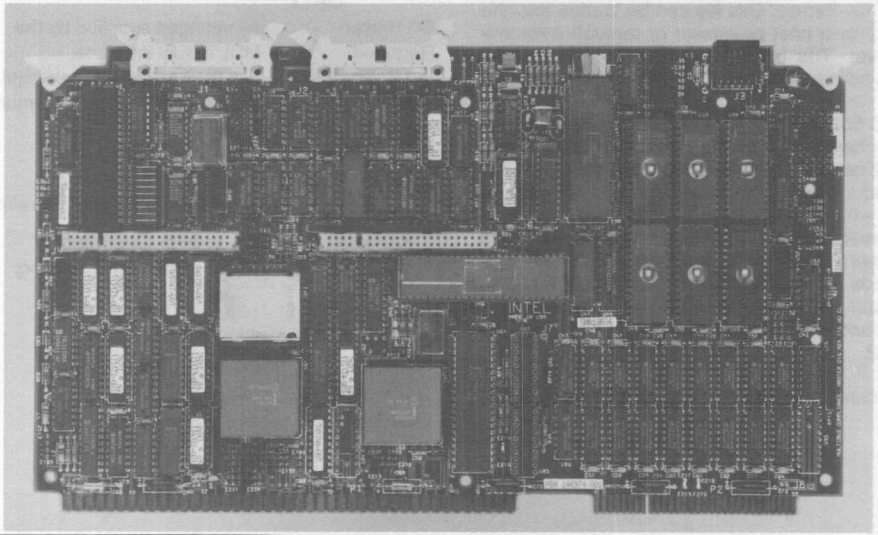
Temperature	0°C to 55°C Operating −40°C to 65°C Non-Operating
Humidity	5% to 90% Operating 5% to 95% Non-Operating

iSBC® 186/51S COMMUNICATING COMPUTER

MEMBER OF THE OpenNET™ PRODUCT FAMILY

- **6 MHz iAPX 186 Microprocessor**
- **128K Bytes of dual-ported RAM expandable on-board to 256K Bytes**
- **82586 Local Communications Controller for CSMA/CD applications and 82501 Ethernet serial interface for Ethernet/IEEE 802.3 specifications**
- **Two serial interfaces, RS-232C and RS-422A/RS-449 compatible**
- **Sockets for up to 192K Bytes of JEDEC 28 pin standard memory devices**
- **Supports transport layer software (iNA 960) and higher layer communications software (such as RMX-NET)**
- **80130 Real-Time Operating System Firmware**
- **Two iSBX™ bus connectors**
- **16M Bytes address range of MULTIBUS®**
- **MULTIBUS® interface for multimaster configurations and system expansion**
- **Supported by a complete family of single board computers, peripheral controllers, digital & analog I/O, memory, packaging and software**

The iSBC® 186/51S COMMUNICATING COMPUTER, the COMMputer™, is a member of Intel's OpenNET family of products, and supports Intel's network software. The COMMputer utilizes Intel's VLSI technology to provide an economical self-contained computer for applicatons in processing and local area network control. The combination of the iAPX 186 Central Processing Unit/80130 Operating System Firmware and the 82586 Local Communications Controller/82501 Ethernet Serial Interface makes it ideal for applications which require both communication and processing capabilities such as networked workstations, factory automation, office automation, communications servers, and many others. The CPU, Ethernet interface, serial communications interface, 128K Bytes of RAM, up to 192K Bytes of ROM, Operating System Firmware, I/O ports and drivers and the MULTIBUS interface all reside on a single 6.75″ X 12.00″ printed circuit board.



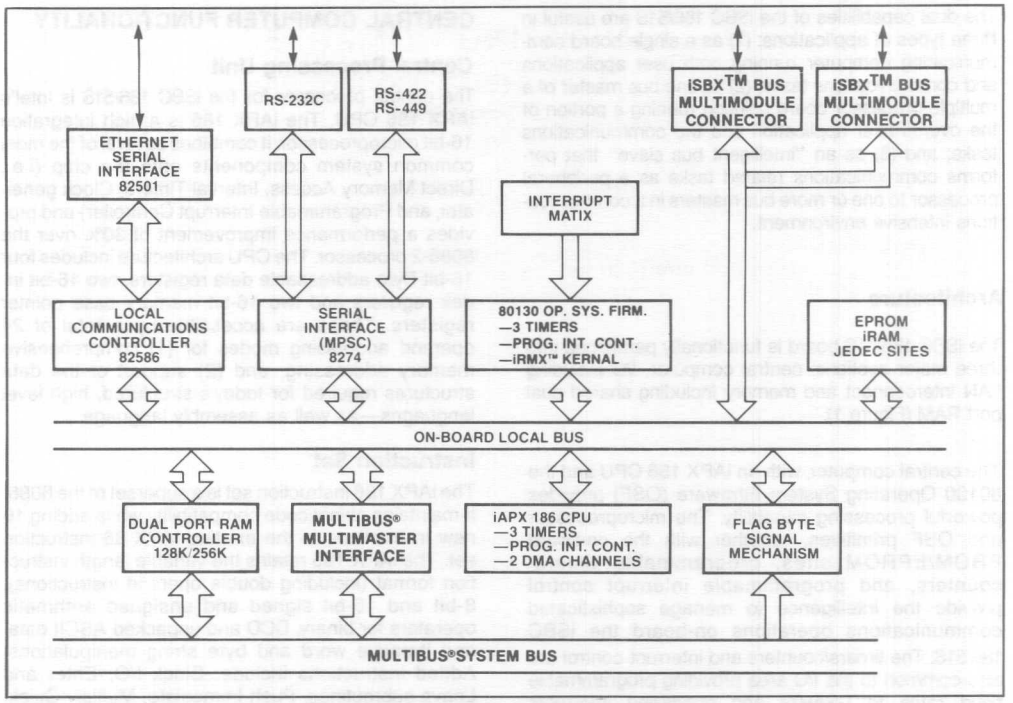


Figure 1. iSBC® 186/51 Block Diagram

FUNCTIONAL DESCRIPTION

Communicating Computer

Intel's OpenNET strategy provides the user with building blocks to implement all seven layers of the International Standards Organization's (ISO) Open Systems Interconnect (OSI) model (see figure 2.) The iSBC 186/51S is a part of the OpenNET product family. The iSBC 186/51S can host iNA 960 transport layer software to provide ISO 8073 class 4 standard protocol on IEEE 802.3 LAN. In conjunction with the transport file access software, RMX-NET, the iSBC 186/51S and iNA 960 provide a complete seven layer communications solution.

The iSBC 186/51S board intergrates a programmable processor and communications capability onto one board, serving both computational and networking capacities as dictated by the application. The communications coprocessor (82586) aids in this task by accomplishing as much of the communications task as possible before the processor intervenes (thus reducing the overhead load of the 80186 processor).

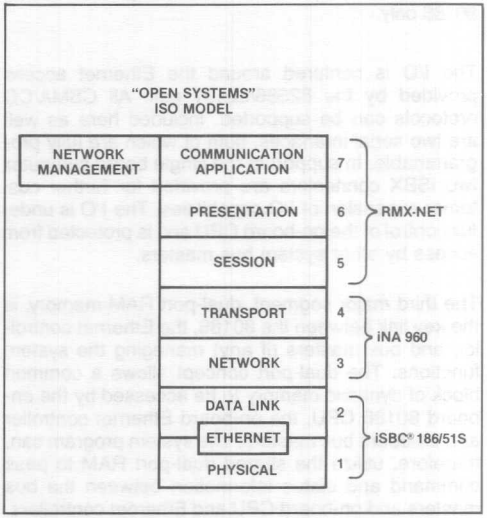


Figure 2. iSBC® 186/51S Impelementation of ISO Standard Model

The dual capabilities of the iSBC 186/51S are useful in three types of applications: (1) as a single board communicating computer running both user applications and communications tasks; (2) as one bus master of a multiple processor board solution running a portion of the overall user application and the communications tasks; and (3) as an "intelligent bus slave" that performs communications related tasks as a peripheral processor to one or more bus masters in a communications intensive environment.

Architecture

The iSBC 186/51S board is functionally partitioned into three major sections: central computer, I/O including LAN interconnect and memory including shared dual port RAM (Figure 1).

The central computer, with an iAPX 186 CPU and the 80130 Operating System Firmware (OSF) provides powerful processing capability. The microprocessor and OSF primitives, together with the on-board PROM/EPROM sites, programmable timers/counters, and programmable interrupt control provide the intelligence to manage sophisticated communications operations on-board the iSBC 186/51S. The timers/counters and interrupt control are also common to the I/O area providing programmable baud rates to USARTs and prioritizing interrupts generated from the USARTs. The central computer functions are protected for access by the on-board 80186 only.

The I/O is centered around the Ethernet access provided by the 82586/82501 pair. All CSMA/CD protocols can be supported. Included here as well are two serial interfaces, both of which are fully programmable. In support of the single board computer, two iSBX connectors are provided for further customer expansion of I/O capabilities. The I/O is under full control of the on-board CPU and is protected from access by other system bus masters.

The third major segment, dual-port RAM memory, is the key link between the 80186, the Ethernet controller, and bus masters (if any) managing the system functions. The dual-port concept allows a common block of dynamic memory to be accessed by the on-board 80186 CPU, the on-board Ethernet controller and off-board bus masters. The system program can, therefore, utilize the shared dual-port RAM to pass command and status information between the bus masters and on-board CPU and Ethernet controllers. In addition, the dual-port concept permits blocks of data transmitted or received to accummulate in the on-board shared RAM, minimizing the need for a dedicated memory board.

CENTRAL COMPUTER FUNCTIONALITY

Central Processing Unit

The central processor for the iSBC 186/51S is Intel's iAPX 186 CPU. The iAPX 186 is a high integration 16-bit microprocessor. It combines several of the most common system components onto the chip (i.e., Direct Memory Access, Interval Timers, Clock generator, and Programmable Interrupt Controller) and provides a performance improvement of 30% over the 8086-2 processor. The CPU architecture includes four 16-bit Byte addressable data registers, two 16-bit index registers and two 16-bit memory base pointer registers. These are accessible by a total of 24 operand addressing modes for (1) comprehensive memory addressing, and (2) support of the data structures required for today's structured, high level languages—as well as assembly language.

Instruction Set

The iAPX 186 instruction set is a superset of the 8086. It maintains object code compatibility while adding 10 new instructions to the existing iAPX 86 instruction set. The iAPX 186 retains the variable length instruction format (including double operand instructions), 8-bit and 16-bit signed and unsigned arithmetic operators for binary, BCD and unpacked ASCII data, and iterative word and byte string manipulations. Added instructions include: Block I/O, Enter and Leave subroutines, Push Immediate, Multiply Quick, Array Bounds Checking, Shift and Rotate by Immediate, and Pop and Push All.

Architectural Features

A six-byte instruction queue provides prefetching of sequential instructions and can reduce the 750 nsec minimum instruction cycle to 250 nsec for queued instructions. The stack oriented architecture readily supports modular programming by facilitating fast, simple, intermodule communication, and other programming constructs needed for asynchronous real-time systems. Using a windowing technique and external logic, the full 16M Bytes addressing range of the IEEE-796 MULTIBUS Standard is available to the user. The dynamic relocation scheme allows ease in segmentation of pure procedure and data for efficient memory utilization. Four segment registers (code, stack, data, extra) contain program loaded offset values which are used to map 16-bit addresses to 20-bit addresses. Each register maps 64K Bytes at a time and activation of a specific register is controlled, both explicitly by program control, and implicitly by specific functions and instructions. A flag byte signaling mechanism aids in creating an interprocessor communication scheme. This includes (1) the ability to set/reset interrupts with MULTIBUS commands and (2) board reset.

OPERATING SYSTEM FUNCTIONALITY

Operating System Firmware

The 80130 provides a set of multitasking kernel primitives, kernel control storage, and the additional support hardware, including system timers and interrupt controller, required by those primitives. To the applications programmer, the OSF extends the iAPX 186 architecture by providing 35 operating system primitive instructions, and supporting five new system data types. This makes the OSF a logical and easy to use architectural extension to the iAPX 186 system design. The chip has also been designed to be compatible with the iRMX86 operating system.

Architecture

The 80130 is connected directly to the local bus of the 80186 processor with address decoding, buffering, and bus-demultiplexing logic contained on-chip (Figure 1). Internally, the 80130 firmware consists of two sections: an operating system unit and a control unit. The former consists of a 16K Byte operating-system-kernel control store complete with an operating-system timer, a delay timer, a bit-rate generator, and 8259A-compatible programmable interrupt logic.

The first timer generates the fundamental real-time clock period in the system. It is set to 10 milliseconds initially but can be modified by the system designer. The delay timer supports the kernel timing function by indicating the next event. Both these timer resources are reserved for use by the kernel.

The bit-rate generator, which has a range of 75 to 768 kilobits per second, is provided as a user resource. The 80130 interrupt logic vectors eight independent priority levels, one of which is reserved for the operating-system timers.

Operation

The 80130 supplements the 80186's basic architecture with five new objects, or system data types: jobs, tasks, segments, mailboxes, and regions. See Tables 1 and 2 for the new data types and operating system primitives.

The 80130 operates by creating, manipulating and deleting individual system objects. When an object is created, the 80130 returns its name to the creating task. This name is referred to and used as an abstract data type, called a TOKEN. The TOKEN is a highly efficient way of accessing the iSBC 186/51S address space. Referring to a segment object, for example, causes a 16-bit address to be loaded into one of the processor segment registers, which can then be used to directly address a paragraph (16-Byte unit) anywhere in the 1M Byte address space. Task creation is also accomplished in this manner and requires only the specification of a priority, a task private data segment (if needed), a task stack, and a task program starting address.

To take full advantage of multiprogramming, the operating system must provide each application with a separate environment—that is, separate memory and tasks. This isolation both protects independent programs from interfering with one another and allows the application programmer to work without regard to the other application programs in the system. The 80130 supports multiprogramming with the job data type. The creation of a job requires the specification of a large number of parameters and is normally done only when the system is being initialized.

Table 1. System Data Types Used in 80130 Operating System Firmware

Job	Jobs are the means of organizing the program environment and resources. An application consists of one or more jobs. Each iAPX 186 system data type is contained in some job. Jobs are independent of each other, but they may share access to resources. Each job has one or more tasks, one of which is an initial task. Jobs are given pools of memory, and they may create subordinate offspring jobs, which may borrow memory from their parents.
Task	Tasks are the means by which computations are accomplished. A task is an instruction stream with its own execution stack and private data. Each task is part of a job and is restricted to the resources provided by its job. Tasks may perform general interrupt handling as well as other computational functions. Each task has a set of attributes, maintained for it by the iAPX 186, which characterize its status. These attributes are: its containing job its register context its priority (0-255) its execution state (asleep, suspended, ready, running, asleep/suspended) its suspension depth its user-selected exception handler its option 8087 extended task state
Segment	Segments are the units of memory allocation. A segment is a physically contiguous sequence of 16-Byte, 8086 paragraph-length, units. Segments are created dynamically from the free memory space of a job as one of its tasks request memory for its use. A segment is deleted when it is no longer needed. The iAPX 186 maintains and manages free memory in an orderly fashion, it obtains memory space from the pool assigned to the containing job of the requesting task and returns the space to the job memory pool (or the parent job pool) when it is no longer needed. It does not allocate memory to create a segment if sufficient free memory is not available to it; in that case it returns an error exception code.
Mailbox	Mailboxes are the means for intertask communication. Mailboxes are used by tasks to send and receive message segments. The iAPX 186 creates and manages two queues for each mailbox. One of these queues contains message segments sent to the mailbox but not yet received by any task. The other mailbox queue consists of tasks that are waiting to receive messages. The iAPX 186 assures that waiting tasks receive messages as soon as messages are available. Thus at any moment one or possibly both of two mailbox queues will be empty.
Region	Regions are the means of serialization and mutual exclusion. Regions are familiar as "critical code regions." The iAPX 186 region data type consists of a queue of tasks. Each task waits to execute in mutually exclusive code or to access a shared data region, for example to update a file record.
Tokens	The OSF interface makes use of a 16-bit TOKEN data type to identify individual OSF data structures. Each of these (each instance) has its own unique TOKEN. When a primitive is called, it is passed the TOKENs of the data structures on which it will operate.

Table 2. 80130 Operating System Firmware Primitives

JOB	CREATE JOB	Creates a job partition including memory pool, task list, and stack area.
TASK	CREATE TASK	Creates a task with the specified environment and priority and puts it in the ready state. Checks for insufficient memory available within the containing job.
	DELETE TASK	Deletes a task from the system as well as from any queues in which it is waiting. The task's state and stack segment are de-allocated.
	SUSPEND TASK	Suspends a task (changes its status to suspended) or increases the task's suspension count by 1. A sleeping task may also be suspended and will then awaken suspended unless resumed.
	RESUME TASK	Decreases the suspension count of a task by 1. If the count is at that point reduced to 0, the task state is made ready or if it was suspend-asleep, it is put back to asleep.
	SLEEP	Puts the task in the asleep state, a number of 10-ms units may be specified.
INTERRUPT	SET PRIORITY	Changes the task's priority to the value passed in the primitive.
	SET INTERRUPT	Assigns an interrupt handler to a level. The task that makes this call is made the interrupt task for the same level, unless the call indicates there is no interrupt task.
	RESET INTERRUPT	Disables an interrupt level. Cancels the interrupt handler, deletes the interrupt task for that level if assigned.
	GET LEVEL	Returns the number of the interrupt level for highest priority interrupt handler currently in operation (several interrupt handlers could be operating).
	EXIT INTERRUPT	Completes interrupt processing and sends end-of-interrupt signal to hardware.
	SIGNAL INTERRUPT	Invokes the interrupt task assigned to a level from that level's interrupt handler.
	WAIT INTERRUPT	Makes the interrupt task state suspended pending a signal interrupt from an interrupt handler. Used by an interrupt task to signal its readiness to service an interrupt.
	ENABLE	Enables an external interrupt level.
	DISABLE	Disables an external interrupt level.
	GET EXCEPTION HANDLER	Reads the location and exception-handling mode of the current operating system exception handler for a task.
	SET EXCEPTION HANDLER	Establishes the location and exception-handling mode of the current operating system exception handler for a task.

Table 2. 80130 Operating System Firmware Primitives (Cont.)

SEGMENT	CREATE SEGMENT	Allocates dynamically an area of memory of a specified length in 16-Byte paragraph units up to a maximum of 64K Bytes (for example, for use as a buffer). Returns a location token for the segment allocated.
	DELETE SEGMENT	De-allocates the memory segment indicated by the parameter token.
	ENABLE DELETION	Allows the system data type value indicated by the location token to be deleted.
	DISABLE DELETION	Prevents the system data type value indicated by the location token from being deleted.
MAILBOX	CREATE MAILBOX	Creates a mailbox with the specified task queueing discipline. Returns a location token.
	DELETE MAILBOX	Deletes a mailbox, and returns its memory. If tasks are waiting for the mailbox, they are awakened (their state is made ready) with an appropriate exception condition. If messages are waiting for tasks, they are discarded.
	SEND MESSAGE	Sends a message segment to a mailbox.
	RECEIVE MESSAGE	A task is ready to receive a message at a mailbox. The task is placed on the mailbox task queue. The task may optionally wait for a response indefinitely, or a number of time intervals (generally 10 ms long), or not at all. When complete, the primitive returns to the task the location token of the message segment received.
REGION	CREATE REGION	Creates a region data type value specifying a queueing discipline. Returns a token for the region.
	DELETE REGION	Deletes a region if and only if the region is not in use.
	ACCEPT CONTROL	Gains control of a region if it is immediately available, but does not wait if it is not available.
	RECEIVE CONTROL	Is the same primitive as accept control but the task that performs it may elect to wait.
	SEND CONTROL	Relinquishes a region.

Programmable Timers

The 80186 provides three internal 16-bit programmable timers. Two of these are highly flexible and are connected to four external pins (two per timer). They can be used to count external events, time external events, generate nonrepetitive waveforms, etc. The third timer is not connected to any external pins, and is useful for real-time coding and time delay applications. In addition, this third timer can be used as a prescaler to the other two, or as a DMA request source. The factory default configuration for timer 0 is baud rate generator.

The 80130 provides three more programmable timers. One is a factory default baud rate generator and outputs an 8254 compatible square wave to the RS232 Channel B. The other two timers are assigned to the use of the OSF and should not be altered by the user.

The system software configures each timer independently to select the desired function. Examples of available functions are shown in Table 3. The contents of each counter may be read at any time during system operation.

Interrupt Capability

The iSBC 186/51S has two programmable interrupt controllers (PICs): one in the 80186 component and one in the 80130 component. In the iRMX mode, the 80186 interrupt controller acts as a slave to the 80130. The 80186 interrupt controller in this mode uses all of its external interrupt pins. It therefore services only internally generated interrupts (i.e., three timers, two DMA channels). The 80130 interrupt controller operates in the master mode and has eight prioritized inputs that can be programmed either edge or level sensitive.

The iSBC 186/51S board provides 9 vectored interrupt levels. The highest level is the NMI (Non-Maskable Interrupt) line which is directly tied to the 80186 CPU. This interrupt is typically used for signaling catastrophic events (e.g., power failure). The Programmable Interrupt Controllers (PIC) provide control and vectoring for the next eight interrupt levels. As shown in Table 4, a selection of four priority processing modes is available for use in designing request processing configurations to match system requirements for efficient interrupt servicing with minimal latencies. Operating modes and priority assignments may be reconfigured dynamically via software at any time during system operation. The PIC accepts interrupt requests from all on-board I/O resources and from the MULTIBUS system bus. The PIC then resolves requests according to the selected mode and, if appropriate, issues an interrupt to the CPU.

Interrupt Request Generation

iSBC 186/51S Interrupt Service requests may originate from 25 sources. Table 5 contains a list of devices and functions supported by interrupts. All interrupts are jumper configurable with either suitcase or wire wrap to the desired interrupt request level.

Table 3. 80186 Programmable Timer Functions

Function	Operation
Interrupt on terminal count	When terminal count is reached, an interrupt request is generated. This function is extremely useful for generation of real-time clocks.
Programmable one-shot	Output goes low upon receipt of an external trigger edge or software command and returns high when terminal count is reached. This function is retriggerable.
Rate generator	Divide by N counter. The output will go low for one input clock cycle, and the period from one low going pulse to the next is N times the input clock period.
Square-wave rate generator	Output will remain high until 1/2 the count has been completed, and go low for the other half of the count.
Software triggered strobe	Output remains high until software loads count (N). N periods after count is loaded, output goes low for one input clock period.
Hardware triggered strobe	Output goes low for one clock period N counts after rising edge counter trigger input. The counter is retriggerable.
Event counter	On a jumper selectable basis, the clock input becomes an input from the external system. CPU may read the number of events occurring after the counter "window" has been enabled or an interrupt may be generated after N events occur in the system.

Table 4. iSBC® 186/51 Programmable Interrupt Modes

Mode	Operation
Fully nested	Interrupt request line priorities fixed at 0 as highest, 7 as lowest.
Special fully nested	Allows multiple interrupts from slave PICs to the master PIC. Used in the case of cascading where the priority has to be conserved within each slave.
Specific priority	System software assigns lowest priority level. Priority of all other levels based in sequence numerically on this assignment.
Polled	System software examines priority-encoded system interrupt status via interrupt status register.

Table 5. Interrupt Request Sources

Device	Function	Number of Interrupts
MULTIBUS® interface	Requests from MULTIBUS® resident peripherals or other CPU	2
8274	Transmit buffer empty, receive buffer full and channel errors	8
Internal 80186 PIC	Timer 0, 1, 2 outputs (function determined by timer mode) and 2 DMA channel interrupts	5
82586 LCC	Communications processor needs attention	1
Flag byte interrupt	Flag byte interrupt set by MULTIBUS master	1
Systick	80130, RMX system timer	1
Edge to level trigger	Converts EDGE interrupts to level interrupts	1
iSBX™ connectors MULTIMODULE™	Function determined by iSBX™	4 (2 per iSBX™ connector)
Bus fail safe timer	Indicates addressed MULTIBUS® resident device has not responded to command within 6 msec	1
OR-gate matrix	Outputs of OR-gates on-board for multiple interrupts	1

I/O FUNCTIONALITY

Local Communications Controller

The 82586 is a local communications controller designed to relieve the iAPX 186 of many of the tasks associated with controlling a local network. The 82586 provides most of the functions normally associated with the data link and physical link layers of a local network architecture. In particular, it performs framing (frame boundary delineation, addressing, and bit error detection), link management, and data modulation. It also supports a network management interface.

The iAPX 186 and the 82586 communicate entirely through a shared memory space. To the user, the 82586 appears as two independent but communicating units: the Command Unit (CU) and the Receive Unit (RU). The CU executes the commands given by

the iAPX 186 to the 82586. The RU handles all activities related to packet reception, address recognition, CRC checking, etc. The two are controlled and monitored by the CPU via a shared memory structure called the System Control Block (SCB). Commands for the CU and RU are placed into the SCB by the host processor. Status information is placed into the SCB by the CU and RU (via the CU). The Channel Attention and Interrupt lines are used by the CPU and the 82586 to get the other to look into the SCB. See Figure 3.

The 82586 features a high level diagnostic or maintenance capability. It automatically gathers statistics on CRC errors, frame alignment errors, overrun errors, and frames lost due to lack of reception resources. In addition, the user can output the status of all internal registers to facilitate system design.

Upon initialization, the 82586 obtains the address of its System Control Block through the Initialization Root which begins at location 0FFFFF6H. See Figure

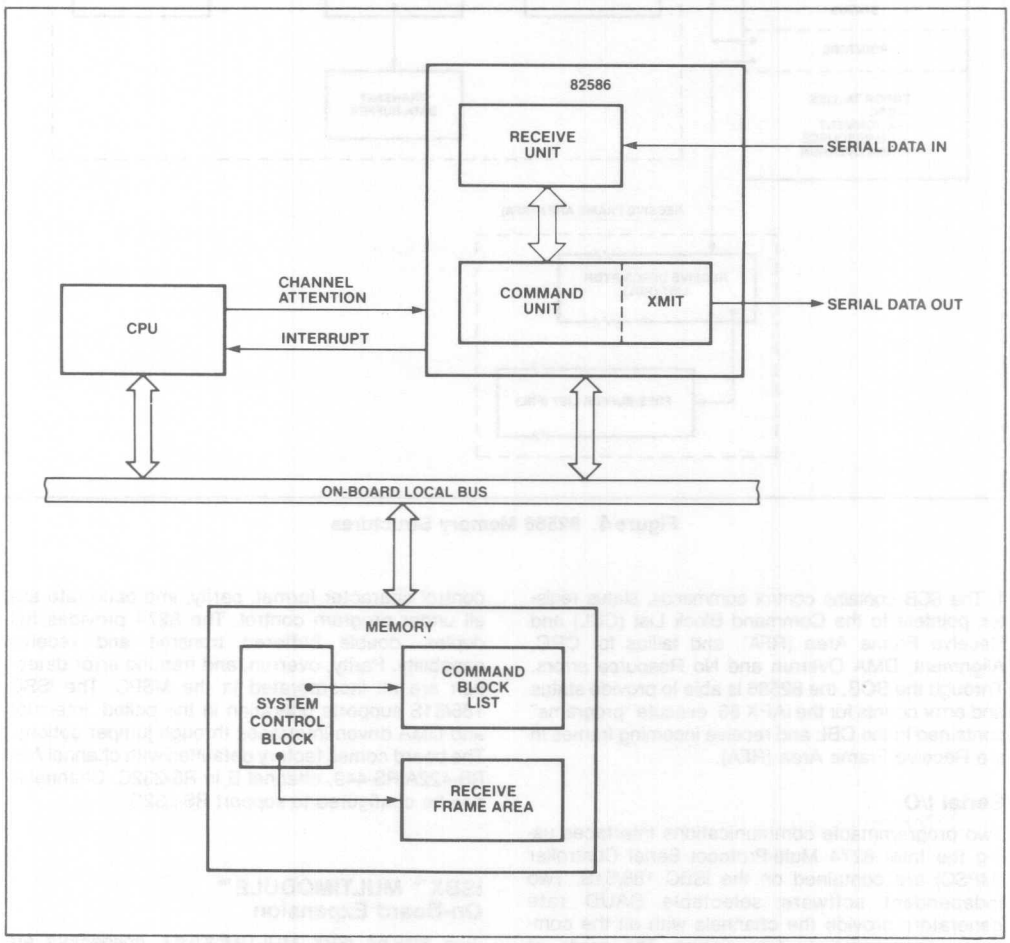


Figure 3. System Overview

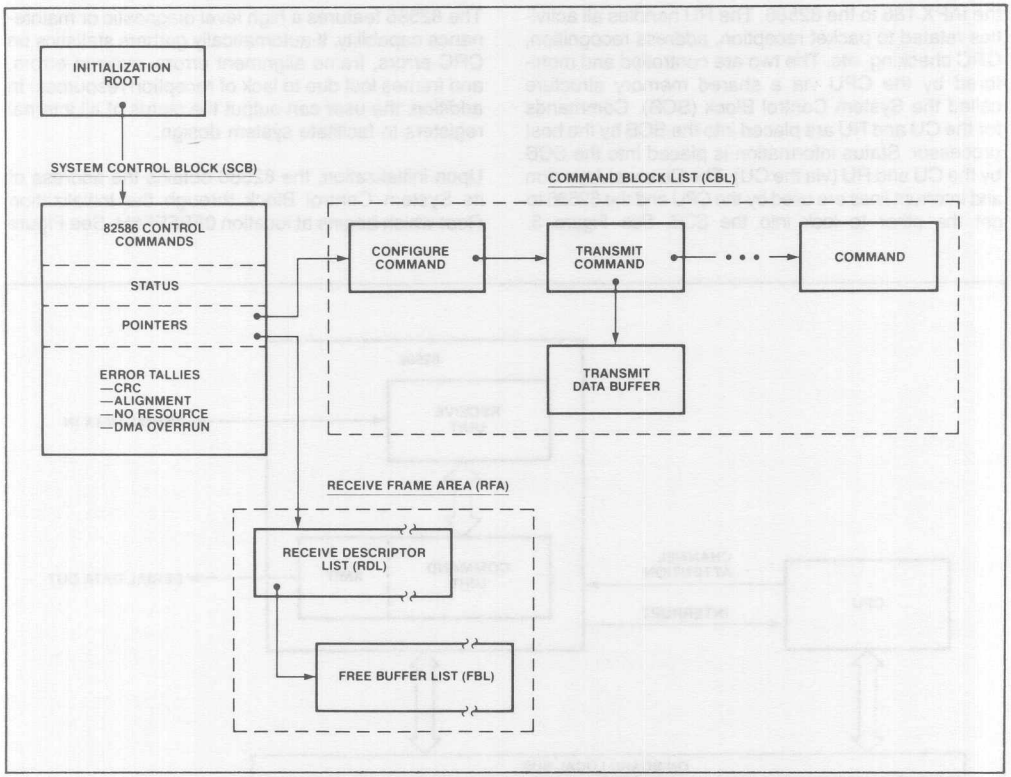


Figure 4. 82586 Memory Structures

4. The SCB contains control commands, status register, pointers to the Command Block List (CBL) and Receive Frame Area (RFA), and tallies for CRC, Alignment, DMA Overrun and No Resource errors. Through the SCB, the 82586 is able to provide status and error counts for the iAPX 86, execute "programs" contained in the CBL and receive incoming frames in the Receive Frame Area (RFA).

Serial I/O

Two programmable communications interfaces using the Intel 8274 Multi-Protocol Serial Controller (MPSC) are contained on the iSBC 186/51S. Two independent software selectable BAUD rate generators provide the channels with all the common communications frequencies. The mode of operation (for example, Asynchronous, Byte Synchronous or Bisynchronous protocols), data format, control character format, parity, and baud rate are all under program control. The 8274 provides full duplex, double buffered transmit and receive capability. Parity, overrun, and framing error detection are all incorporated in the MSPC. The iSBC 186/51S supports operation in the polled, interrupt and DMA driven interfaces through jumper options. The board comes factory defaulted with channel A in RS-422A/RS-449, channel B in RS-232C. Channel A can be configured to support RS-232C.

iSBX™ MULTIMODULE™ On-Board Expansion

Two 8/16-bit iSBX MULTIMODULE connectors are provided on the iSBC 186/51S microcomputer. Through these connectors, additional on-board I/O functions

may be added. iSBX MULTIMODULE boards optimally support functions provided by VLSI peripheral components such as additional parallel and serial I/O, analog I/O, small mass storage device controllers (e.g., cassettes and floppy disks), and other custom interfaces to meet specific needs. By mounting directly on the single board computer, less interface logic, less power, simpler packaging, higher performance, and lower cost results when compared to other alternatives such as MULTIBUS form factor compatible boards. The iSBX connectors on the iSBC 186/51S boards provide all signals necessary to interface to the local on-board bus, including 16 data lines for maximum data transfer rates. iSBC MULTIMODULE boards designed with 8-bit data paths and using the 8-bit iSBX connector are also supported on the iSBC 186/51S microcomputers. A broad range of iSBX MULTIMODULE options are available in this family from Intel. Custom iSBX modules may also be designed for use on the iSBC 186/51S boards. An iSBX bus interface specification and iSBX connectors are available from Intel.

MEMORY FUNCTIONALITY

RAM Capabilities

The iSBC 186/51S COMMputer board contains 128K Bytes of dual-port dynamic RAM. The on-board RAM may be expanded to 256K Bytes with the iSBC 304 MULTIMODULE board mounted onto the iSBC 186/51S board. The dual-port controller allows access to the on-board RAM (including RAM MULTIMODULE options) from the iSBC 186/51S board and from any other MULTIBUS master via the system bus. Segments of on-board RAM may be configured as a private resource, protected from MULTIBUS system access. The amount of memory allocated as a private resource may be configured in increments of 25% of the total on-board memory ranging from 0% to 100% (optional RAM MULTIMODULE board doubles the increment size). These features allow the multiprocessor systems to establish local memory for each processor and shared system memory configurations where the total system memory size (including local on-board memory) can exceed one megabyte without addressing conflicts.

Universal Memory Sites for Local Memory

Six 28-pin sockets are provided for the use of Intel's 2732, 2764, 27128, 27256 EPROMs and their respective ROMs. When using the 27256s, the on-board EPROM capacity is 192K Bytes. Other JEDEC standard pinout devices are also supported, including byte-wide static RAMs and iRAMs.

MULTIBUS® SYSTEM BUS AND MULTIMASTER CAPABILITIES

Overview

The MULTIBUS system bus is Intel's industry standard microcomputer bus structure. Both 8 and 16-bit single board computers are supported on the MULTIBUS structure with 24 address and 16 data lines. In its simplest application, the MULTIBUS system bus allows expansion of functions already contained on a single board computer (e.g., memory and digital I/O). However, the MULTIBUS structure also allows very powerful distributed processing configurations with multiple processors and intelligent slave I/O, and peripheral boards capable of solving the most demanding microcomputer applications. The MULTIBUS system bus is supported with a broad array of board level products, LSI interface components, detailed published specifications and application notes.

Expansion Capabilities

Memory and I/O capacity may be expanded and additional functions added using Intel MULTIBUS compatible expansion boards. Memory may be expanded by adding user specified combinations of RAM boards, EPROM boards, or combination boards. Input/output capacity may be added with digital I/O and analog I/O expansion boards. Mass storage capability may be achieved by adding single or double density diskette controllers, or hard disk controllers. Modular expandable backplanes and cardcages are available to support multiboard systems.

Multimaster Capabilities

For those applications requiring additional processing capacity and the benefits of multiprocessing (i.e., several CPU's and/or controllers logically sharing system tasks through communication of the system bus), the iSBC 186/51S boards provide full MULTIBUS arbitration control logic. This control logic allows up to *three iSBC 186/51S boards* or other bus masters, including iSBC 80 family MULTIBUS compatible 8-bit single board computers to share the system bus using a serial (daisy chain) priority scheme. This allows up to 16 masters to share the MULTIBUS system bus with an external parallel priority decoder. In addition to the multiprocessing configurations made possible with

multimaster capability, it also provides a very efficient mechanism for all forms of DMA (Direct Memory Access) transfers.

MISCELLANEOUS FUNCTIONALITY

Power-Fail Control and Auxiliary Power

An active-low TTL compatible memory protect signal is brought out on the auxiliary connector which, when asserted, disables read/write access to RAM memory on the board. This input is provided for the protection of RAM contents during system power-down sequences. An auxiliary power bus is also provided to allow separate power to RAM for systems requiring battery back-up of read/write memory. Selection of this auxiliary RAM power bus is made via jumpers on the board.

System Development Capabilities

The development cycle of iSBC 186/51S products can be significantly reduced and simplified by using either the System 86/3XX or the Intellec Series Microcomputer Development Systems. The Assembler, Locating Linker, Library Manager, Text Editor and System Monitor are all supported by the ISIS-II disk-based operating system. To facilitate conversion of the 8080A/8085A assembly language programs to run on the iSBC 186/51S boards, CONV-86 is available under the ISIS-II operating system.

In-Circuit Emulator

The Integrated Instrumentation In-Circuit Emulator (I^2ICE) provides the necessary link between the software development environment provided by the Intellec system and the "target" iSBC 186/51S execution system. In addition to providing the mechanism for loading executable code and data into the iSBC 186/51S boards, the I^2ICE-186 provides a sophisticated command set to assist in debugging software and final integration of the user hardware and software.

PL/M-86 and C-86

Intel has two systems implementation languages, PL/M-86 and C-86. Both are standard in the System 86/3XX and are also available as Intellec Microcomputer Development System options. PL/M-86 provides the capability to program in algorithmic language and eliminates the need to manage register usage or allocate memory while still allowing explicit control of the system's resources when needed. C-86 is especially appropriate in applications requiring portability and code density. FORTRAN 86 and PASCAL 86 are also available on Intellec or 86/3XX systems.

Run-Time Support

Intel also offers two run-time support packages: iRMX 88 Realtime Multitasking Executive and the iRMX 86 Operating System. The iRMX 88 executive is a simple, highly configurable and efficient foundation for small, high performance applications. Its multitasking structure establishes a solid foundation for modular system design and provides task scheduling and management, intertask communication and synchronization, and interrupt servicing for a variety of peripheral devices. Other configurable options include terminal handlers, disk file system, debuggers and other utilities. The iRMX 86 Operating System is a highly functional operating system with a very rich set of features and options based on an object-oriented architecture. In addition to being modular and configurable, functions beyond the nucleus include a sophisticated file management and I/O system, and a powerful human interface. Both packages are easily customized and extended by the user to match unique requirements.

SPECIFICATIONS

Word Size

Instruction—8, 16, 24, or 32 bits
Data—8, 16 bits

System Clock

6.00 MHz ± 0.1%

Cycle Time

Basic Instruction Cycle

6 MHz—1000ns
333ns (assumes instruction in the queue)

Note: Basic instruction cycle is defined as the fastest instruction time (i.e., two clock cycles.)

Memory Response Time

	Max Access Time	Min Cycle Time
RAM	—	750ns
Universal Memory Sites	200ns	500ns
Jumper Selectable	300ns	625ns

Memory Capacity/Addressing

Six Universal Memory Sites support JEDEC 24/28 pin EPROM, PROM, iRAM and static RAM.

Example for EPROM:

Device	Total Capacity	Address Range
2732	24K Bytes	$F8000\text{-}FFFFF_H$
2764	48K Bytes	$F0000\text{-}FFFFF_H$
27128	96K Bytes	$E0000\text{-}FFFFF_H$
27256	192K Bytes	$C0000\text{-}FFFFF_H$

On-Board RAM

Board	Total Capacity	Address Range
iSBC 186/51	128K Bytes	$0\text{-}1FFFF_H$

With MULTIMODULE™ RAM

Board	Total Capacity	Address Range
iSBC 304	256K Bytes	$0\text{-}3FFFF_H$

I/O Capacity

Serial—two programmable channels using one 8274
iSBX™ Multimodule™—two 8/16-bit iSBX™ connectors allow use of up to 2 single-wide modules or 1 single-wide module and 1 double-wide iSBX module.

Serial Communications Characteristics

Synchronous —5-8 bit characters; internal or external character synchronization; automatic sync insertion
Asynchronous —5-8 bit characters; break character generation; 1, 1/2, or 2 stop bits; false start bit detection

Baud Rates

Frequency (KHz) (S/W Selectable)	Baud Rate (Hz)		
	Synchronous	Asynchronous	
	÷1	÷16	÷64
153.6	—	9600	2400
76.8	—	4800	1200
38.4	38,400	2400	600
19.2	19,200	1200	300
9.6	9,600	600	150
4.8	4,800	300	75
2.4	2,400	150	—
1.76	1,760	110	2400

NOTE:
Frequency selected by I/O write of appropriate 16-bit frequency factor to baud rate register (80186 timer 0 & 80130 baud timer).

Timers

Input Frequencies

Reference 1.5 MHz ± 0.1% (.5μSec period nominal)
Event Rate: 1.5 MHz max.

80186 Output Frequencies/Timing Intervals

Function	Single Timer/Counter		Dual (Cascaded) Timer/Counter	
	Min	Max	Min	Max
Real-time Interrupt	667ns	43.69ms	667ns	47.72 minutes
Programmable one-shot	1000ns	43.69ms	1000ns	47.72 minutes
Rate generator	22.889 Hz	1.5 MHz	.0003492 Hz	1.5 MHz
Square-wave rate generator	22.889 Hz	1.5 MHz	.0003492 Hz	1.5 MHz
Software triggered strobe	1000ns	43.69ms	1000ns	47.72 minutes
Event counter	—	1.5 MHz	—	—

Interfaces

Ethernet—IEEE 802.3 compatible
MULTIBUS®—IEEE 796 compatible
MULTIBUS®—Master D16 M24 I16 V0 EL

Compliance

iSBX™ Bus—IEEE P959 compatible
Serial I/O—RS-232C compatible, configurable as a data set or data terminal, RS-422A/RS-449

Connectors

Interface	Double-Sided Pins	Centers (in.)	Mating Connectors
Ethernet	10	0.1	AMP87531-5
MULTIBUS® SYSTEM	86 (P1)	0.156	Viking 3KH43/9AMK12 Wire Wrap
	60 (P2)	0.1	Viking 3KH30/9JNK
iSBX™ Bus 8-Bit Data	36	0.1	iSBX™ 960-5
16-Bit Data	44	0.1	iSBX™ 960-5
Serial I/O	26	0.1	3M 3452-0001 Flat or AMP88106-1 Flat

Physical Characteristics

Width—12.00 in. (30.48 cm)
Height—6.75 in. (17.15 cm)
Depth—0.70 in. (1.78 cm)
Weight—18.7 ounces

Environmental Characteristics

Operating Temperature—0°C to 55°C
Relative Humidity—10% to 90% (without condensation)

Electrical Characteristics

DC Power Supply Requirements

Configuration	Maximum Current (All Voltages ± 5%)		
	+5	+12	−12
SBC 186/51S as shipped:			
Board Total	7.45A	40mA	40mA
With separate battery back-up	6.30A	40mA	40mA
Battery back-up	1.15A	—	—
With SBC-304 Memory Module Installed:			
Board Total	7.55A	40mA	40mA
With separate battery back-up	6.30A	40mA	40mA
Battery back-up	1.25A	—	—

NOTES:
1. Add 150 mA to 5V current for each device installed in the 6 available Universal Memory Sites.
2. Add 500 mA to 12V current if Ethernet transceiver is connected.
3. Add additional currents for any SBX modules installed.

Reference Manual

122136-001—iSBC 186/51 Hardware Reference Manual (NOT SUPPLIED)

Manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.

ORDERING INFORMATION

Part Number	Description
SBC 186/51S	Communicating Computer

iNA 960 TRANSPORT SOFTWARE

MEMBER OF THE OpenNET™ PRODUCT FAMILY

- ISO Transport (8073) Class 4 services
  - —Guaranteed message integrity
  - —Data rate matching (flow control)
  - —Multiple connection capability
  - —Variable length messages
  - —Expedited delivery
  - —Negotiation of virtual circuit characteristics during opens
- Additional functionality
  - —Connectionless transport (Datagram)
  - —External Data Link
- IEEE 802.3 Data Link protocol (CSMA/CD) supported
- Comprehensive Network Management services
  - —Collection of network usage statistics
  - —Setting and inspecting of transport and data link parameters
  - —Fault isolation and detection
  - —Boot Server
- Compatible with multiple system environments
  - —Runs as an iRMX™ 86 job
  - —Supports host operating system independent designs based on 8086, 8088 or 80168 and 82586 components
- Runs on iSBC® 186/51 COMMputer™ Board
- Preconfigured version runs on SXM 552 Transport Engine

iNA 960 is a general purpose local area network software package implementing the class 4 services of the ISO transport specification and network management functions in system designs based on the 8086, 8088 and 80186 microprocessors and the 82586 communications co-processor. iNA 960 also supports Intel's board level LAN products, the iSBC® 552, iSXM™ 552, and the iSBC®186/51. Combined with these board products iNA 960 provides a cost effective, high performance industry standard transport capability supporting the OpenNET higher layer software or other user application.

iNA 960 is a ready-to-use software building block for OEM suppliers of networked systems for both technical and commercial applications. Examples for such applications include networked design stations, manufacturing process control, communicating word processors, and financial services workstations. Using the iNA 960 software the OEM can minimize development cost and time while achieving compatibility with a growing number of equipment suppliers adapting the IEEE and ISO standards.

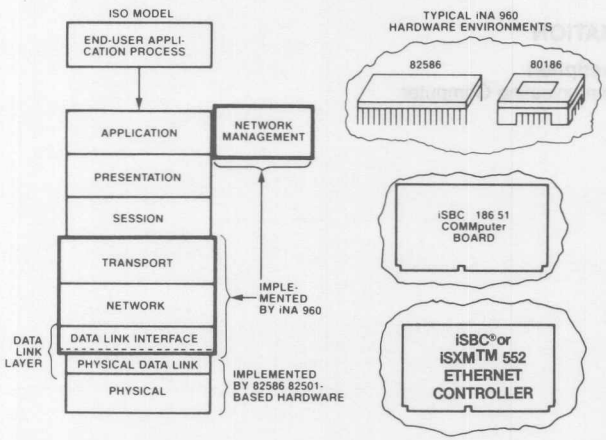


Figure 1.

FUNCTIONAL OVERVIEW

The iNA 960 design is a standard implementation of the Class 4 transport protocol defined by the ISO OSI model. The Transport Layer provides a reliable full-duplex message delivery service on top of the "best effort" IEEE 802.3 standard packet delivery service implemented by the 82586 (or equivalent) physical and data link functions.

Consisting of linkable modules, the software can be configured to implement a range of capabilities and interface protocols. In addition to reliable process-to-process message delivery, the capabilities include a datagram service, a boot server, a direct user access to the Data Link Layer, and a comprehensive network management facility.

iNA 960 can be configured to run under iRMX 86 along with the user software, or to run on top of a dedicated 8086, 8088 or 80186 processor coupled with an 82586 to provide a communications front end processor.

The software also includes a Network Management service. This facility enables the user to monitor and adjust the network's operation in order to optimize its performance.

The current release of iNA 960 includes a "null" Network Layer supporting the Data Link and Transport Layers without providing internetwork routing service. This capability will be implemented in later releases of iNA 960.

For a conceptual block diagram of iNA 960, refer to Figure 2.

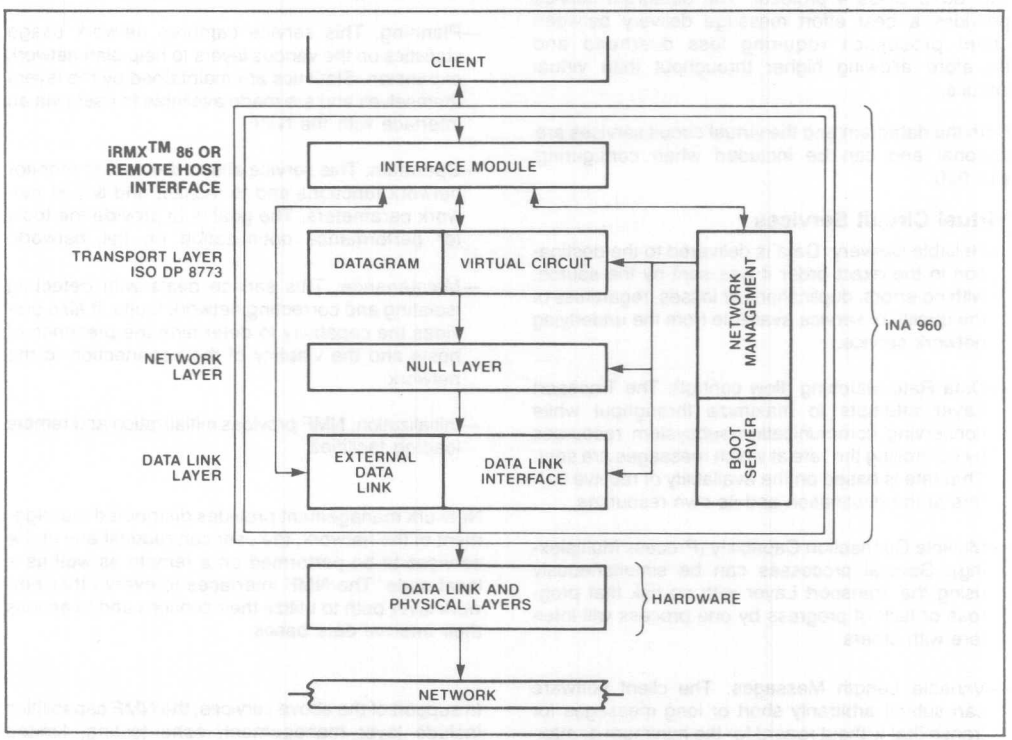


Figure 2. iNA 960 Conceptual Block Diagram

TRANSPORT LAYER

The Transport Layer provides message delivery services between client processes running on computers (network "hosts" or "nodes") anywhere in the network.

Client processes are identified by a combination of a network address defining the node and a transport service access point defining the interface point through which the client accesses the transport services. The combined parameters, called the transport address, are supplied by the user for both the local and the remote client processes to be connected.

The iNA 960 transport layer implements two kinds of message delivery services: virtual circuit and datagram. The virtual circuit provides a reliable point-to-point message delivery service ensuring maximum data integrity, and it is fully compatible with the ISO 8073 Class 4 protocol. The datagram service provides a best effort message delivery between client processes requiring less overhead and therefore allowing higher throughput than virtual circuits.

Both the datagram and the virtual circuit services are optional and can be included when configuring iNA 960.

Virtual Circuit Services

—Reliable Delivery: Data is delivered to the destination in the exact order it was sent by the source, with no errors, duplications or losses, regardless of the quality of service available from the underlying network service.

—Data Rate Matching (flow control): The Transport Layer attempts to maximize throughput while conserving communication subsystem resources by controlling the rate at which messages are sent. That rate is based on the availablity of receive buffers at the destination and its own resources.

—Multiple Connection Capability (Process Multiplexing): Several processes can be simultaneously using the Transport Layer with no risk that progress or lack of progress by one process will interfere with others.

—Variable Length Messages: The client software can submit arbitrarily short or long messages for transmittal without regard for the minimum or maximum network service data unit (NSDU) lengths supported by the underlying network services.

—Expedited Delivery (optional). With this service the client can transmit up to 16 bytes of urgent data bypassing the normal flow control. The expedited data is guaranteed to arrive before any normal data submitted afterward.

Connectionless Transport (Datagram) Service

The datagram service transfers data between client processes without establishing a virtual circuit. The service is a "best effort" capability and data may be lost or misordered. Data can be transferred at one time to a single destination or to several destinations (multicast).

NETWORK MANAGEMENT FACILITY (NMF)

The network management facility provides the users of the network with planning, operation, maintenance and initialization services described below.

—Planning: This service captures network usage statistics on the various layers to help plan network expansion. Statistics are maintained by the layers themselves and are made available to users via an interface with the NMF.

—Operation: This service allows the user to monitor network functions and to inspect and adjust network parameters. The goal is to provide the tools for performance optimization on the network.

—Maintenance: This service deals with detecting isolating and correcting network faults. It also provides the capability to determine the presence of hosts and the viability of their connection to the network.

—Initialization: NMF provides initialization and remote loading facilities.

Network management provides distributed management of the network; the user can request any of the services to be performed on a remote as well as a local node. The NMF interfaces to every other network layer both to utilize their services and to access their internal data bases.

In support of the above services, the NMF capabilities include layer management, echo testing, limited debugging facilities, and the ability to down line load and dump a remote system.

Layer management deals with manipulating the internal database of a layer. The elements of these data bases are termed objects. Some examples for objects are the number of collisions, retransmission time-out limit, the number of packets sent, and the list of nodes to boot. NMF can examine and modify objects in a layer's data base.

An echo facility is provided. Using this facility the host can determine if a node is present on the network or not, test the communication path to that node and determine whether the remote node is functional.

NMF enables the user to read or write memory in any host present on the network. This feature is provided as an aid to debugging.

NMF can down line load any system present on the network. A simple Data Link protocol is used to ensure reliability. This facility can be used to load databases, to boot systems without local mass storage or to boot a set of nodes remotely, thus ensuring that they have the same version of software, etc.

Dumping is an operation equivalent to memory read from the user's standpoint; however, dumping uses the Data Link facilities while memory read uses the transport facilities.

EXTERNAL DATA LINK (EDL)

The External Data Link option allows the user to access the functionalities of the Data Link Layer directly instead of having to go through the network and transport layers. This flexibility is useful when the user needs custom higher layer software, or does not need the Network Layer and Transport Layer services (e.g., when sending "best effort" messages, or running customer diagnostics).

Through the EDL the capabilities supporting the lower layers in iNA 960 are made directly available to the user. EDL enables the user to establish and delete data link connections, transmit packets to individual and multiple receivers, and configure the data link software to meet the requirements of the given network environment.

USER ENVIRONMENT

iNA 960 is designed to run on hardware based on the 8086, 8088 or 80186 microprocessors and the 82586 LAN Coprocessor. The software can be configured to run under iRMX 86 or on a dedicated 8086, 8088 or 80186 processor separately from the host. The following section describes these two operating environments.

iRMX™ Environment

In this configuration, both the user program and iNA 960 are running under iRMX 86. The communications software is implemented as an iRMX 86 job requiring the nucleus only for most operations. The only exception is the boot server option which also needs the Basic I/O System. iNA 960 will run in any iRMX environment including configurations based on the 80130. See Figure 3 and 4 for an illustration of iNA 960 running under iRMX 86.

Operating System/Processor Independent Implementation

In those systems where iRMX 86 is not the primary operating system, where off-loading the host of the communications tasks is necessary for performance reasons, or where an existing communications front-end processor configuration is being upgraded, the user may wish to dedicate a processor for communications purposes. iNA 960 can be configured to support such implementations by providing network services on an 8086, 8088 or 80186 processor. Figure 5 depicts the conceptual block diagram of this configuration. The SBC & SXM 552 are MULTIBUS® implementations of this architecture.

This approach provides the component and system designer with an ISO standard communications software building block that can be adapted to his system's needs with a minimum interfacing effort. For added flexibility, iNA 960 provides the user with the alternative of using the included interface module or writing his own module, if necessary.

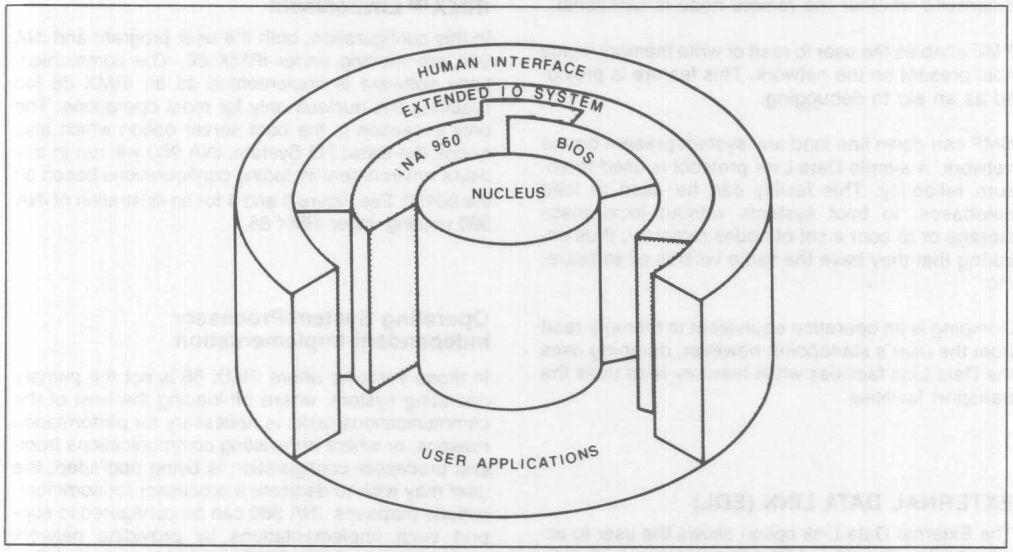


Figure 3. As an iRMX™ job, iNA 960 uses nucleus calls and, when the Boot Server is present, BIOS calls.

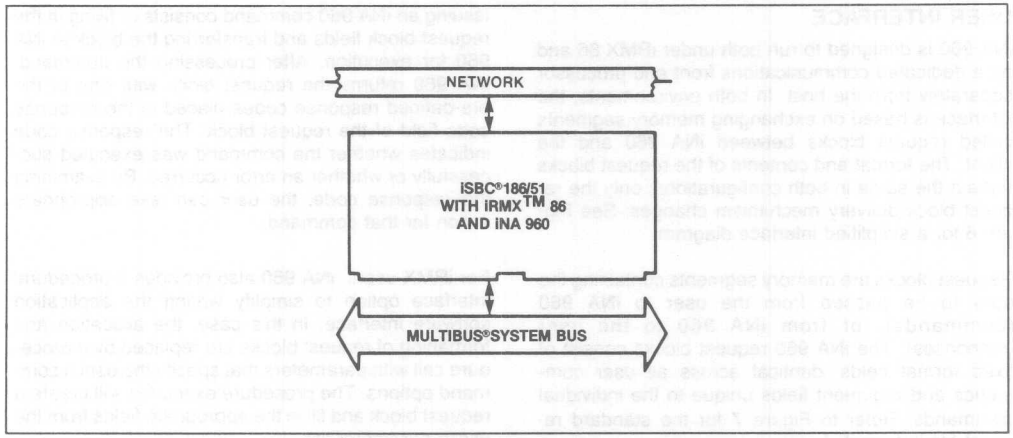


Figure 4. Configuration using iSBC® 186/51, iRMX™ 86 and iNA 960.

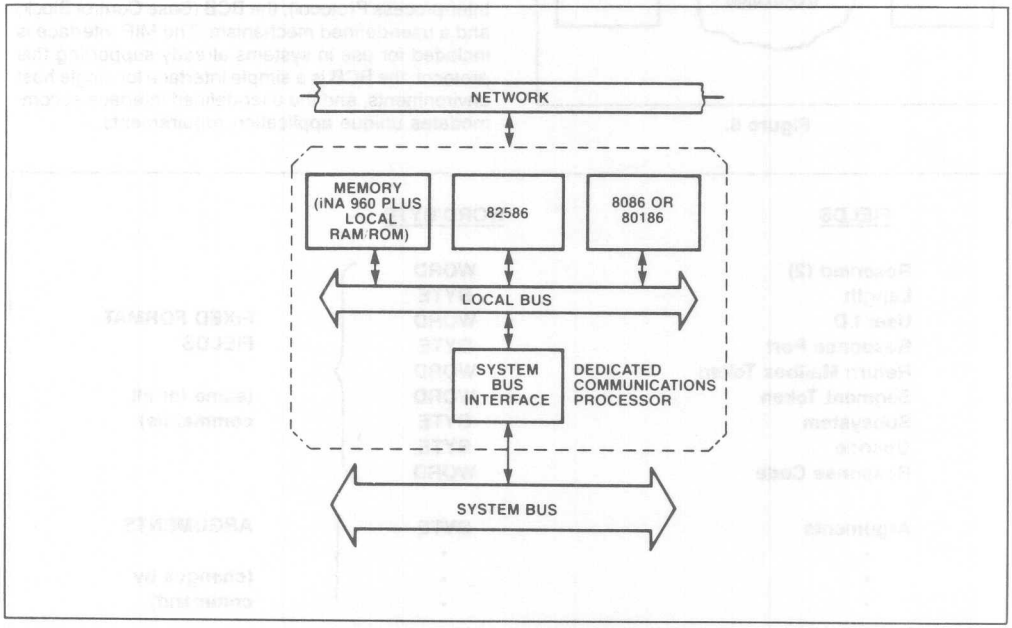


Figure 5. In the operating system/processor independent implementation iNA 960 is running on a dedicated 8086, 8088 or 80186 processor.

USER INTERFACE

iNA 960 is designed to run both under iRMX 86 and on a dedicated communications front end processor separately from the host. In both environments, the interface is based on exchanging memory segments called request blocks between iNA 960 and the client. The format and contents of the request blocks remain the same in both configurations; only the request block delivery mechanism changes. See Figure 6 for a simplified interface diagram.

Request blocks are memory segments containing the data to be passed from the user to iNA 960 (commands), or from iNA 960 to the user (responses). The iNA 960 request blocks consist of fixed format fields identical across all user commands and argument fields unique to the individual commands. Refer to Figure 7 for the standard request block format.

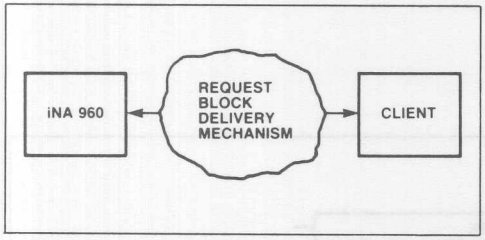


Figure 6.

Issuing an iNA 960 command consists of filling in the request block fields and transferring the block to iNA 960 for execution. After processing the command, iNA 960 returns the request block with one of the pre-defined response codes placed in the response code field of the request block. The response code indicates whether the command was executed successfully or whether an error occurred. By examining the response code, the user can take appropriate action for that command.

For iRMX users, iNA 960 also provides a procedural interface option to simplify writing the application software interface. In this case, the allocation and formatting of request blocks are replaced by a procedure call with parameters that specify the user's command options. The procedure execution will create a request block and fill in the appropriate fields from the user's parameter list.

For component users the request block delivery mechanism is the means by which the host processor and the communications processor running iNA 960 software exchange the request blocks. iNA 960 provides three such mechanisms: the MIP (Multibus Inter-process Protocol), the BCB (Base Control Block) and a user-defined mechanism. The MIP interface is included for use in systems already supporting this protocol; the BCB is a simple interface for single host environments, and the user-defined interface accommodates unique application requirements.

FIELDS	WORD/BYTE	
Reserved (2)	WORD	FIXED FORMAT FIELDS (same for all commands)
Length	BYTE	
User I.D.	WORD	
Response Port	BYTE	
Return Mailbox Token	WORD	
Segment Token	WORD	
Subsystem	BYTE	
Opcode	BYTE	
Response Code	WORD	
Arguments	BYTE	ARGUMENTS (changes by command)
•	•	
•	•	
•	•	

Figure 7. iNA 960 Request Block Format

Transport Layer User Interface

The following table summarizes the user commands and the corresponding transport layer responses.

Command	Function
1. OPEN	Allocates memory for the connection data base of a virtual circuit (or connection) to be established. The connection database contains data concerning the connection.
2. SEND CONNECT REQUEST	Requests connection to a fully specified remote transport address using specified ISO connection negotiation options.
3. AWAIT CONNECT REQUEST TRAN	Indicates that the transport client is willing to consider incoming connection requests based on pre-established acceptance criteria.
4. AWAIT CONNECT REQUEST USER	Indicates that the transport client is willing to consider incoming connection requests. If the request meets the address and negotiation option criteria, it is passed to the client for further consideration.
5. ACCEPT CONNECT REQUEST	Indicates that the connection requested by a remote transport service is accepted by the client.
6. SEND DATA or SEND EOM DATA	With this command, the client requests the transmission of the data in the buffers using the normal delivery service of the specified connection. The SEND EOM DATA command signals that the end of the data marks the end of the transport service data unit.
7. RECEIVE DATA	Posts normal receive data buffers for a specific connection or for a buffer pool used by a class of connections.
8. SEND EXPEDITED DATA	Transmits up to 16 bytes of data using the expedited delivery service. The expedited data is guaranteed to arrive at the destination before any normal data submitted afterward.
9. RECEIVE EXPEDITED DATA	Posts receive data buffers for expedited delivery for a specific connection or for a pool of buffers used by a class of connections.
10. CLOSE	Terminates an existing connection or rejects an incoming connection request. Any normal or expedited data queued up to be sent will not be sent.
11. AWAIT CLOSE	Requests notification of the client of the termination of a specified connection.
12. SEND DATAGRAM	Requests transmission of the data in the buffers using the transport datagram service.
13. RECEIVE DATAGRAM	Posts a receive buffer for a specific receiver or a class of receivers to receive data from a transport datagram.

Network Management Layer User Interface

Command	Function
1. READ OBJECT	Returns the value of the specified object to the client.
2. SET OBJECT	Sets the value of an object as specified by the client.
3. READ AND CLEAR OBJECT	Returns the value of the specified object to the client then clears the object.
4. ECHO	This function is used to determine the presence of a node, to test the communication path to the node and to ascertain the viability and functionality of the remote host addressed.
5. UP LINE DUMP	Requests a remote node to dump a specified memory area.
6. READ MEMORY	Reads memory of the specified network node.
7. SET MEMORY	Sets memory of the specified network node.
8. FORCE LOAD	Causes a node to attempt a remote load from another node.

External Data Link Interface

Command	Function
1. CONNECT	With this command the client establishes a data link connection.
2. DISCONNECT	Eliminates a previously established connection.
3. TRANSMIT	Transmits data contained in buffers specified by the client.
4. POST RECEIVE PACKET DESCRIPTOR	Allocates memory for maintaining records on receive data buffers. Also may be used to allocate memory for buffering receive data.
5. POST RECEIVE BUFFER	Allocates memory for buffering receive data.
6. ADD MULTICAST ADDRESS	Adds an address to the list of data link multicast addresses.
7. REMOVE MULTICAST ADDRESS	Removes an address from the list of data link multicast addresses.
8. SET DATA LINK I.D.	Sets up a unique data link I.D. for the station.

CONFIGURING iNA 960

In order to adapt iNA 960 to his specific application, the user must configure the software to define the desired functions, to select the appropriate interface, to set the layer parameters and to set up for the required hardware configuration.

There are a number of capability combinations the user may elect to implement in his application. At the transport layer level the options are: virtual circuit service with or without expedited delivery, or datagram service, or both. At the data link level, the user may include or exclude the External Data Link interface.

The Network Management Facility is also optional. When it is configured in, the user may also include the boot server module. These capabilities can be made available simply by linking in the corresponding software modules. The interface options are also implemented in a modular fashion; the user links in the desired module to set up for the iRMX 86 or the operating system independent configurations.

Layer parameters and confiuration options are first edited into layer configuration files, then assembled and linked into iNA 960. Layer parameters adjust the network's operation to match the usage pattern and the available resources. For example, within the Transport Layer, the flow control parameters, the retransmission timer parameters, the transport data base parameters, etc. can be set via this process.

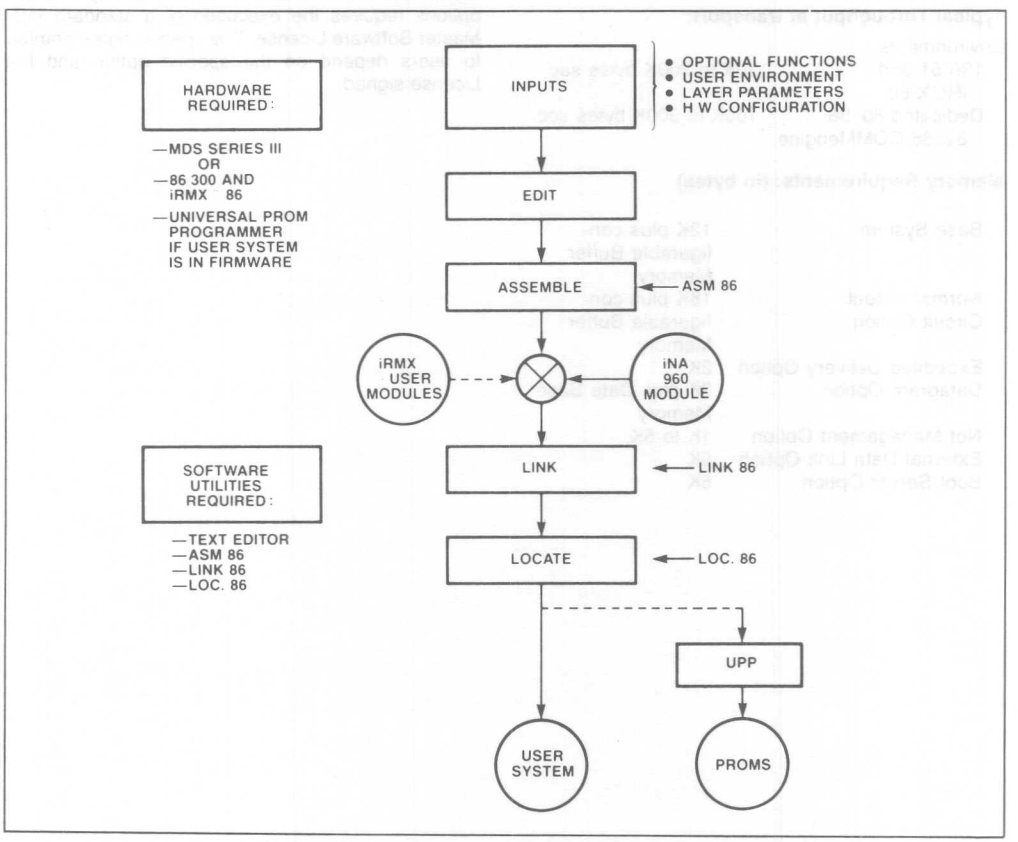


Figure 8. The Configuration Process for iNA 960

The user also sets up for the required hardware configuration, such as port addresses and interrupt levels, during this process. For the flow diagram of configuring iNA 960, refer to Figure 8.

SPECIFICATIONS

Hardware Supported:

—iSBC 186/51 Communicating Computer
—SBC 552 COMMengine
—SXM 552 Transport Engine (runs with preconfigured transport software)
—Custom designs based on 8086, 8088 and 80186 microprocessors and the 82586 Local Communications Controller.

Typical Throughput at transport:

Environments:	
186/51 and iRMX 86	50K to 200K bytes sec
Dedicated 80186 82586 COMMengine	100K to 300K bytes sec

Memory Requirements: (in bytes)

Base System	12K plus configurable Buffer Memory
Normal Virtual Circuit Option	18K plus configurable Buffer Memory
Expedited Delivery Option	2K
Datagram Option	3K plus Data Base Memory
Net Management Option	1K to 5K
External Data Link Option	5K
Boot Server Option	5K

Available Literature/ Reference Materials:

—iNA 960 Programmer's Reference Manual (11/83)
—iSBC 186/51 Data Sheet (Now)
—iSBC 186/51 Hardware Reference Manual (11/83)

ORDERING INFORMATION

The following is a list of ordering options for the iNA 960 Transport Software. All options include a full year of update service that provides a periodic NEWSLETTER, Software Problem Report Service, and copies of system updates that occur during this period. All of the object code options listed are available on either ISIS or RMX compatible double density diskettes.

As with all Intel software, purchase of any of these options requires the execution of a standard Intel Master Software License. The specific rights granted to users depend on the specific option and the License signed.

Order Code	Description
iNA 960 YRO	OEM object code license requiring the payment of incorporation fees for each derivative work based on iNA 960; ISIS and RMX formatted diskettes
iNA 960 YST	Object code license to use the product at a second site or facility; ISIS and RMX formatted diskettes
iNA 960 YBY	Object code buy-out license requiring no further payment of incorporation fees; ISIS and RMS formatted diskettes
iNA 960 YSU	Object code single use license only; ISIS and RMS formatted diskettes
iNA 960 ESR	License for machine readable source code if iNA 960. RMX and 51V formatted diskettes
iNA 960 LST	Source listing of iNA 960 provided on microfiche under a special source code license agreement
iNA 960 RF	Order code for the payment of incorporation fees

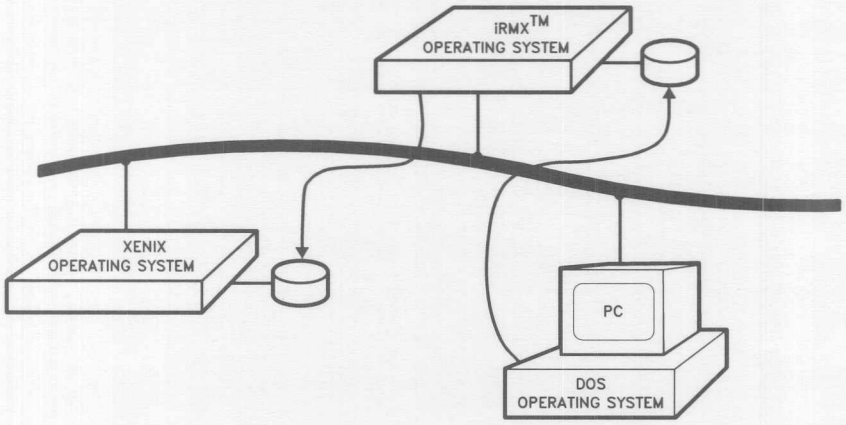
iRMX™ NETWORKING SOFTWARE-iRMX™-NET

MEMBER OF THE OpenNET™ PRODUCT FAMILY

- **Transparent Network File Access**
  - **Remote files can be worked with as if they were local**
- **Connects iRMX™, XENIX* and DOS systems on the LAN****
  - **Compatible with XENIX Networking Software (XENIX* NET) and MS-NET/ IBM PC Networking program**
- **Runs under iRMX™ 86 Operating System**
- **Existing applications can be distributed without change**
- **Supports OpenNET™—Ethernet hardware and software**
  - **iSXM™ 552 Transport Engine**
  - **iSBC® 552 COMMengine**
  - **iSBC® 186/51 COMMputer™**
  - **iNA 960 Transport software**
- **Supports file server applications**
  - **Based on iRMX™ 86 Basic I/O system**
- **Distributed name server**

The Intel OpenNET™ iRMX™ Network File access software provides transparent file access between iRMX and XENIX* and iRMX and MS/DOS systems across a LAN. Users can use local file systems commands to read, write, open, close, etc. files residing at remote iRMX, MS/ or PC/DOS and XENIX systems. IRMX NET implements the upper layer ISO OSI protocols used by the IBM PC Network Program and XENIX NET. Interoperation among these systems is supported by Intel's LAN product line including the iSXM 552 Transport engine, the iSBC® 552 COMMengine, the iSBC® 186/51 COMMputer™ and the iNA 960 Transport software. Networked iRMX systems serve in a wide range of applications including real time transactions, automated testing, data collection, communications switching, etc.

*XENIX is a trademark of Microsoft Corp.
**RMX to XENIX interoperation will be fully qualified only in R1.1 and up.



231372-1

iRMX™-NET FUNCTIONAL DESCRIPTION

iRMX™-NET provides transparent remote file access capability through a file consumer and a file server module. The consumer intercepts file commands from the local user and transmits them across the LAN to the server at the node where the target file resides. The server receives, interprets and executes the command acting as a user to its local file system. The user has the option of configuring either or both in his target system.

RMX-NET also includes a name server which provides name-to-address mapping. The iRMX-NET file consumer uses the name server to find the physical address of the referenced system.

The capabilities allow iRMX systems to interoperate over the LAN with XENIX systems configured with XENIX-NET or DOS systems using MS-NET or IBM PC Network Program. This interoperation entails accessing data and loading programs through the network, sharing common servers and communication between users.

The network file service requires the support of an underlying ISO 8073 compatible transport service provided by the iNA 960 network software running on the iSBC 186/51 COMMputer or the iSXM 552/ iSBC 552 boards. In terms of the ISO OSI reference model iRMX-NET, in conjunction with the transport service and Ethernet/IEEE 802.3 hardware, provide complete seven layer functionality and serves as the fundamental building block for the development of a host of other services such as mail or virtual terminal (see Figure 1).

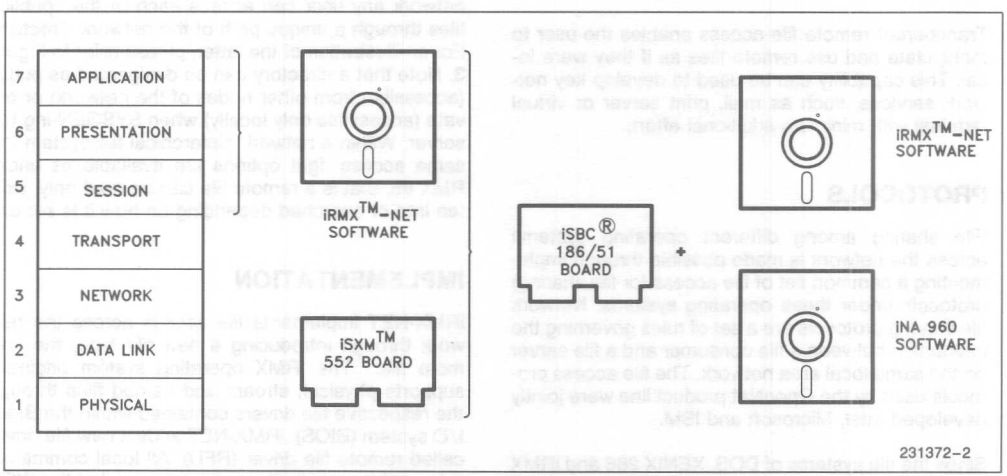


Figure 1. ISO OSI Reference Model RMX-NET

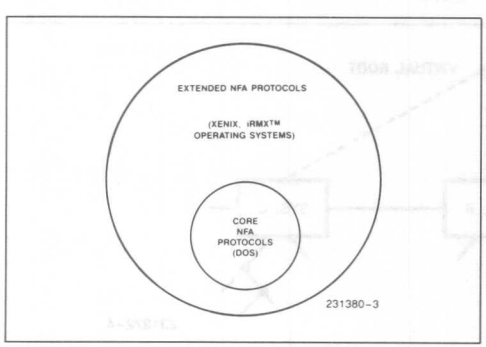


Figure 2. Protocols and Interoperation

CONSUMER	SERVER	WHICH PROTOCOL USED	SUPPORTED IN
iRMX™	iRMX™	EXT.	R1.0
iRMX™	XENIX	EXT.	R1.1
XENIX	iRMX™	EXT.	R1.1
XENIX	XENIX	EXT.	R1.0
XENIX	DOS	CORE	R1.0
DOS	iRMX™	CORE	R1.0
DOS	XENIX	CORE	R1.0
DOS	DOS	CORE	M/S NETWORK, IBM PC NETWORK SOFTWARE

Table 1. Protocols

TRANSPARENT REMOTE FILE ACCESS

iRMX-NET provides transparent remote file access at the BIOS, EIOS and Human Interface level. This means that all iRMX 86 applications written using BIOS, EIOS or HI commands can be used in a networked environment where the referenced files may reside at other nodes of the network.

With Release 1 of RMX-NET the user (file consumer) can transparently access files resident at remote systems configured with iRMX-NET or XENIX-NET (file servers). On the other hand, an RMX file server supports remote nodes configured with iRMX-NET, XENIX-NET, Microsoft Networks and IBM PC Network Software file consumers. For a table showing the combinations supported with the initial OpenNET product line please refer to Figure 2.

Transparent remote file access enables the user to manipulate and use remote files as if they were local. This capability can be used to develop key network services, such as mail, print server or virtual terminal with minimum additional effort.

PROTOCOLS

File sharing among different operating systems across the network is made possible through implementing a common set of file access (or file sharing) protocols under these operating systems. Network file sharing protocols are a set of rules governing the interaction between a file consumer and a file server on the same local area network. The file access protocols used by the OpenNet product line were jointly developed Intel, Microsoft and IBM.

Since the file systems of DOS, XENIX 286 and iRMX 86 are not identical, two protocol sets have been devised to support transparency in the various server-consumer combinations. The so-called "core protocols" support transparent file access between two DOS nodes on the network. The "extended protocols" support transparent file access between iRMX and XENIX nodes. The extended protocols contain the core protocols as a subset. See Figure 2 for an illustration. The core and extended protocols are in public domain and can be implemented under other operating systems, thus enabling a host of otherwise incompatible systems to share data and resources and to communicate across the network.

NETWORK HIERARCHICAL FILE SYSTEM

The file sharing protocols implemented in a network extend the file systems of the individual nodes into a so-called network hierarchical file system. Within a network any user can access each of the "public" files through a unique path of the network directory. For an illustration of the latter, please refer to Figure 3. Note that a directory can be designated as public (accessible from other nodes of the network) or private (accessible only locally) when SYSGEN-ing the server. Within a network hierarchical file system the same access right options are available as under RMX 86, that is a remote file can be read only, written into or searched depending on how it is set up.

IMPLEMENTATION

iRMX-NET implements file access across the network through introducing a new file type, the "remote file." The iRMX operating system originally supports physical, stream and named files through the respective file drivers contained within the Basic I/O system (BIOS). iRMX-NET adds a new file driver called remote file driver (RFD). All local commands referencing remote files are intercepted at the BIOS level and are redirected through the RFD to the network.

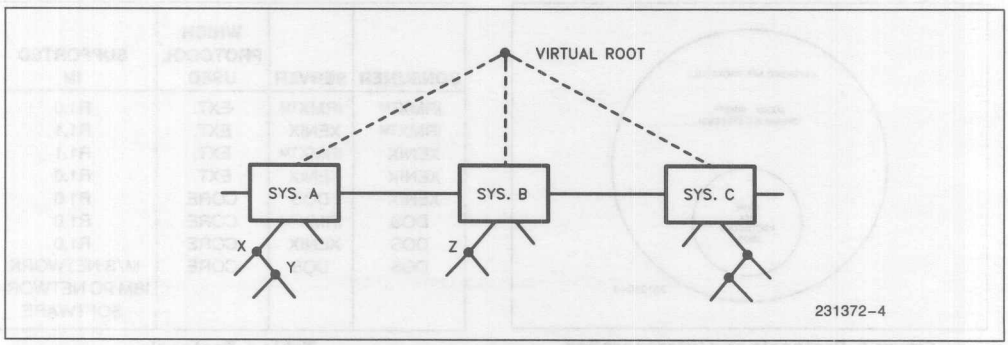


Figure 3. Network Hierarchical File System

The server receives the command from the network and forwards it to the local operating system acting as a user for the local file system. For an implementation block diagram please refer to Figures 4 and 5.

The consumer consists of two basic building blocks. The RFD is operating system dependent and must be configured to run under the host. The file consumer building block is supported by the special executive of iNA 960 and can run on a separate processor along with iNA 960.

The server includes a file server building block and a name server module which are configured to run with iNA 960 and are operating system independent. The server interfaces to the host operating system through the File Access interface which runs under the host operating system.

NAME SERVER

The Name Server provides name to network address mapping for the users. iRMX-NET implements a distributed or "protocol based" name server scheme in which every node "knows" its own name and address and thus there is no "master directory" file within the system.

When a user is referencing a remote node on the network by its name the file consumer broadcasts a request for that name across the network. The only node having the name called will respond by sending its address to the requestor.

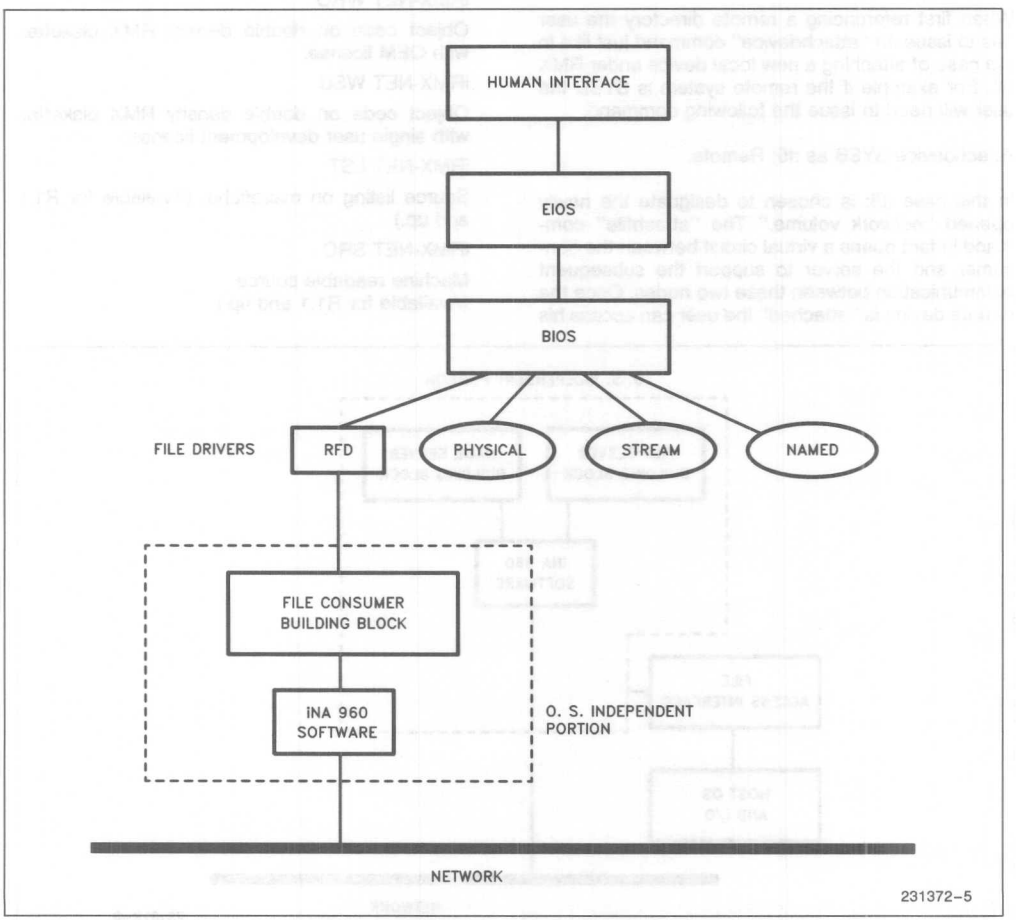


Figure 4. iRMX-NET File Consumer Implementation

SYSTEM ENVIRONMENT

iRMX-NET is supported by any system in which iRMX 86 is at release level 6.0 or later and in which the iNA 960 transport software is already configured in.

iRMX-NET is included at sysgen time as a first level job if the extended I/O system is not present or as an I/O job if it is present. iRMX-NET contains a number of user-defined parameters which must be set up when configuring the system, These parameters include the size of buffers, the number of consumers served concurrently or the maximum permissible number of outstanding processes.

USING iRMX-NET

When first referencing a remote directory the user has to issue an "attachdevice" command just like in the case of attaching a new local device under RMX 86. For example if the remote system is SYSB the user will need to issue the following command:

Attachdevice SYSB as :f5: Remote.

In this case :f5: is chosen to designate the newly opened "network volume." The "attachfile" command in fact opens a virtual circuit between the consumer and the server to support the subsequent communication between these two nodes. Once the remote device is "attached" the user can access his remote and local files alike. As a file server to a DOS consumer, iRMX-NET functions just like a PC AT file server. As a server to XENIX consumer there are a few limitations to transparency, for example, the "LOCK" and "LINK" XENIX commands are not supported under iRMX. As a file consumer to a XENIX server iRMX-NET provides full transparency.

SPECIFICATIONS

— Code size: about 40 KB

— System requirements: - RMX 86 R6.0 or later
- iNA 960

— Throughput: T. B. D.

ORDERING INFORMATION

iRMX-NET WRO

Object code on double density RMX diskettes with OEM license.

iRMX-NET WSU

Object code on double density RMX diskettes with single user development license.

iRMX-NET LST

Source listing on microfiche. (Available for R1.1 and up.)

iRMX-NET SRC

Machine readable source
(Available for R1.1 and up.)

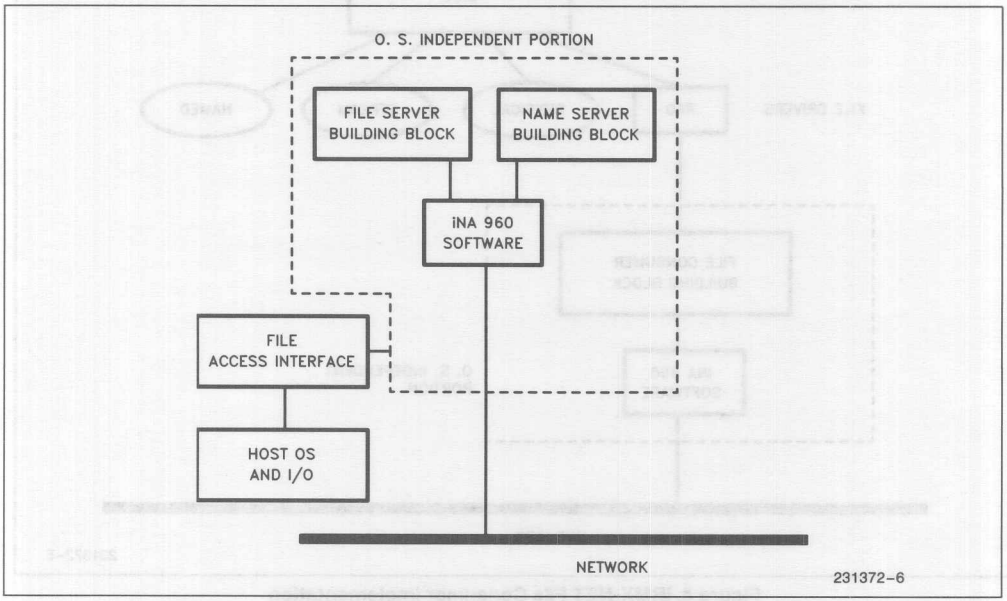


Figure 5. File Server Implementation

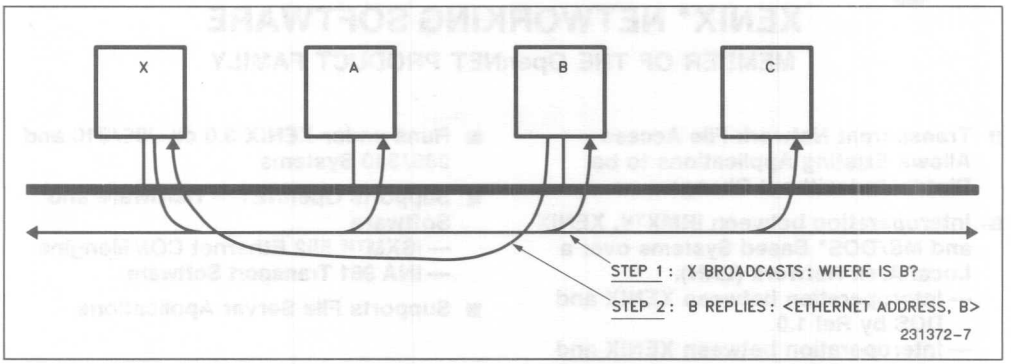


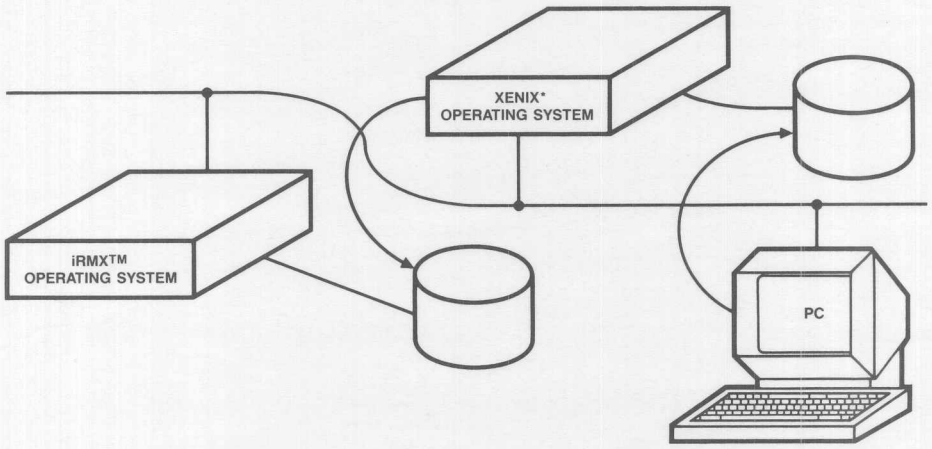
Figure 6. Distributed Name Server Scheme

XENIX* NETWORKING SOFTWARE

MEMBER OF THE OpenNET PRODUCT FAMILY

- **Transparent Network File Access Allows Existing Applications to be Distributed without Change**
- **Interoperation between iRMX™, XENIX and MS/DOS* Based Systems over a Local Area Network (LAN).**
  - **— Interoperation between XENIX and DOS by Rel 1.0**
  - **— Interoperation between XENIX and iRMX by Rel 1.1**
- **Runs under XENIX 3.0 on 286/310 and 286/380 Systems**
- **Supports OpenNET™ Hardware and Software**
  - **— iSXM™ 552 Ethernet COMMengine**
  - **— iNA 961 Transport Software**
- **Supports File Server Applications**

XENIX Networking software is a part of Intel's OpenNET Product Family which provides transparent file access between iRMX, XENIX and MS/DOS systems across a LAN. Users can use local file systems commands to read, write, open, close, etc. files residing at remote iRMX, MS/DOS and XENIX systems. The XENIX Networking Software implements the upper layer protocols used by Microsoft Networks, Interoperation among these systems is supported by Intel's OpenNET LAN product line including the iSXM 552 Transport Engine, iNA 961 (a preconfigured version of iNA 960 transport software), the iSBC® 186/51 COMMputer™ and the iNA 960 Transport software. Networked XENIX systems serve in a wide range of applications, such as distributed data processing, development, scientific and engineering applications, and graphics. Below is a diagram of the OpenNET Local Area Network.



231380-1

* XENIX and MS/DOS are trademarks of Microsoft Corporation.

XENIX—NETWORKING FUNCTIONAL DESCRIPTION

The XENIX Networking software provides transparent remote file access capability through a file consumer and a file server module. The consumer intercepts file commands from the local user application and transmits them across the LAN to the server at a network system or node where the target file resides. The server receives, interprets, and executes the command acting as a user to its local file system. The user has the option of configuring either or both the consumer and server in his target system.

The XENIX Networking Software also includes a name server which allows a logical name to be used to refer to remote nodes instead of the physical LAN address.

The capabilities allow XENIX systems to interoperate over the LAN with RMX systems (with release 1.1) configured with RMX Networking software or MS/DOS systems (with release 1.0) using Microsoft Networks. This interoperation entails accessing data and loading programs through the network, sharing common servers, and communication between users.

The XENIX Networking Software requires the support of an underlying ISO 8073 compatible transport service provided in the iNA 960 network software running on the iSXM 552 Transport Engine. In terms of the ISO OSI reference model, XENIX Networking in conjunction with the transport service and Ethernet hardware provides complete seven layer functionality and serves as the fundamental building block for the development of other services such as mail and remote execution (see Figure 1).

TRANSPARENT REMOTE FILE ACCESS

XENIX Networking provides transparent remote file access at the application interface level. This means that all XENIX 3.0 applications written using operating system file access commands can be used without change in a networked environment where the referenced files may reside at other nodes of the network.

With release 1.0 of the XENIX Networking software, the user (file consumer) can transparently access files resident at remote systems configured with XENIX or MS/DOS file servers. While a XENIX file server supports remote nodes configured with both XENIX and Microsoft Networks and file consumers. For a table showing the combinations supported with the initial OpenNET product line, please refer to Figure 2.

Transparent remote file access enables the user to manipulate and use remote files as if they were local. This capability is used for key network services, such as mail, print server, and remote execution on other XENIX nodes.

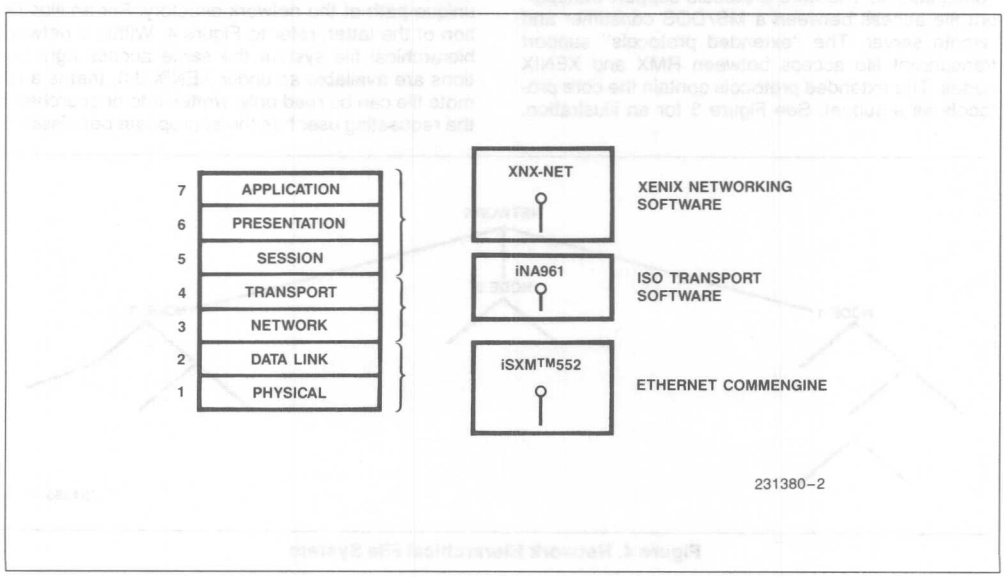


Figure 1. OpenNET™ Product Offerings

Consumer	Server	Protocol Used	Supported In
iRMX™	iRMX™	EXT.	R1.0
iRMX™	XENIX	EXT.	R1.1
XENIX	iRMX™	EXT.	R1.1
XENIX	XENIX	EXT.	R1.0
XENIX	MS/DOS	CORE	R1.0
MS/DOS	iRMX™	CORE	R1.0
MS/DOS	XENIX	CORE	R1.0
MS/DOS	MS/DOS	CORE	MICROSOFT NETWORKS

Figure 2. Interoperation

NETWORK FILE ACCESS PROTOCOLS

File sharing among different operating systems across the network is made possible through implementing a common set of file access (or file sharing) protocols under these operating systems. Network file sharing protocols are a set of rules governing the interaction between a file consumer and a file server on the same local area network. The file access protocols used by the OpenNET product line were jointly developed by Intel, Microsoft, and IBM.

Since the file systems of DOS, XENIX, and iRMX are not identical, two protocol sets have devised to support transparency in the various server-consumer combinations. The core protocols support transparent file access between a MS/DOS consumer and remote server. The "extended protocols" support transparent file access between RMX and XENIX nodes. The extended protocols contain the core protocols as a subset. See Figure 3 for an illustration.

The core and extended protocols are in public domain and can be implemented under other operating systems, thus enabling a host of otherwise incompatible systems to share data resources and to communicate across the network.

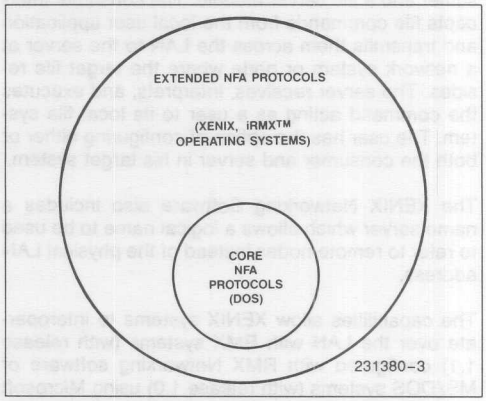


Figure 3. Network File Access Protocols

NETWORK HIERARCHICAL FILE SYSTEM

The file sharing protocols implemeted in a network extend the file systems of the individual nodes into a hierarchical file system. Within a network any user can access each of the "public" files through a unique path of the network directory. For an illustration of the latter, refer to Figure 4. Within a network hierarchical file system the same access right options are available as under XENIX 3.0, that is a remote file can be read only, written into or searched if the requesting user has the appropriate permissions.

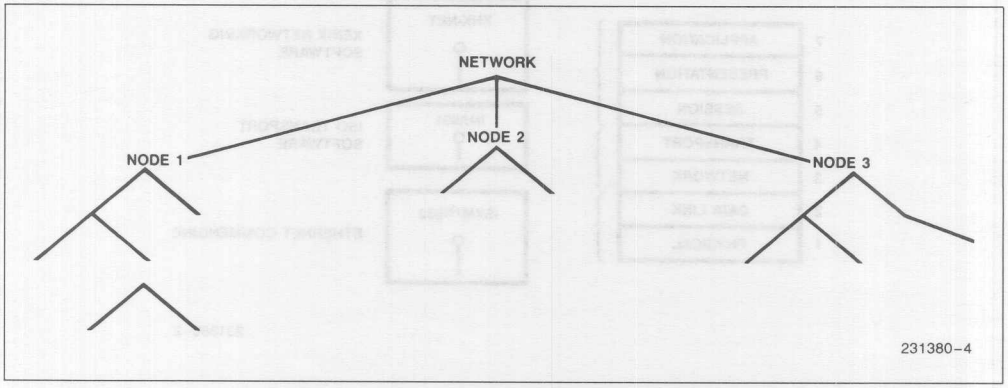


Figure 4. Network Hierarchical File System

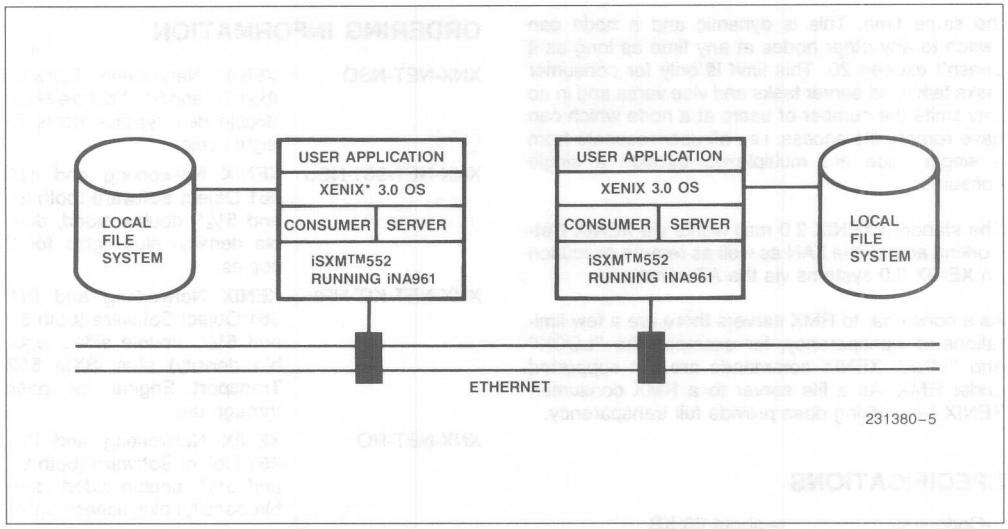


Figure 5. XENIX Networking Consumer/Server

IMPLEMENTATION

The XENIX Networking software implements file access across the network through enhancing the file naming syntax. The logical name associated with a remote system (or node) is appended by the user to the path name of the required file. This nomenclature is distinguished from normal path names by a double slash (//). A similar technique is used for MS/DOS and RMX.

//<node name>/<path name

Hooks have been imbedded in the standard XENIX 3.0 Operating System offered by Intel which detect remote file accesses. XENIX Networking consists of a consumer task and server task. All local commands referencing remote files are intercepted at the kernel level and are redirected through the consumer software to the network.

The server software receives the command from the network and forwards it to the local operating system acting as a user for the local file system. For an implementation block diagram see Figure 5.

The consumer includes a name server module which is configured to run with the iNA 961 Transport Software and is operating system independent. The name server accesses a local file which keeps track of valid node names and their physical LAN address.

SYSTEM ENVIRONMENT

The XENIX Networking software can be used on any system running Intel's XENIX 3.0. This includes the 286/310 and 286/380 systems. Since networking "hooks" are already included in the operating system nothing other than loading the XENIX Networking software onto the local system is necessary. Special network utilities are included for building and maintaining the network configuration files so that the network can be tailored to meet each customers needs.

The network supports a single community of users which means that a user name is unique across the network and therefore users can log-in at any system on the network.

File security is provided by the standard XENIX 3.0 file protection of owner, group, and other access. A local node can restrict local access for remote users by allowing all, none, or a selected few remote nodes.

USING XENIX NETWORKING

When the networking software and configuration files are located in each node, all each node has to do is start the consumer and/or start its server to make its files available to other network systems to start referencing remote files immediately. Each node can talk to as many as 20 other nodes at

the same time. This is dynamic and a node can switch to any other nodes at any time as long as it doesn't exceed 20. This limit is only for consumer tasks talking to server tasks and vice versa and in no way limits the number of users at a node which can have remote file access, i.e., all user requests from a single node are multiplexed through a single consumer.

The standard XENIX 3.0 mail works via XENIX Networking across the LAN as well as remote execution on XENIX 3.0 systems via the AT command.

As a consumer to RMX servers there are a few limitations to transparency, for example, the "LOCK" and "LINK" XENIX commands are not supported under RMX. As a file server to a RMX consumer, XENIX Networking does provide full transparency.

SPECIFICATIONS

—Code size: —about 60 KB
plus 40 K for buffers

—System requirements —XENIX 3.0

—iNA 961

— XENIX Networking along with the iNA 961 software and the iSXM 552 have been qualified for the 286/310-17, 286/310-41, and 286/380 systems.

ORDERING INFORMATION

XNX-NET-NSO	XENIX Networking Software (both 8″ and 5 ¼ ″ double sided, double density) plus rights for eight copies
XNX-NET-961-NSU	XENIX Networking and iNA 961 Object Software (both 8″ and 5¼″ double sided, double density) plus rights for 8 copies
XNX-NET-KIT-NRI	XENIX Networking and iNA 961 Object Software (both 8″ and 5¼″ double sided, double density) plus iSXM 552 Transport Engine for pass through use
XNX-NET-RO	XENIX Networking and iNA 961 Object Software (both 8″ and 5¼″ double sided, double density) plus license rights
XNX-NET-RF	Software Incorporation Fee
XNX-NET-NSR	Machine Readable source code for the XENIX Networking Software. (both 8″ and 5¼″ double sided, double density)
iNA-961-RO	iNA 961 Transport Software plus license rights
iSXM 552	Ethernet Transport Engine plus one iNA 961 Software Incorporation Fee
SYS 310-41XN	XENIX System 286/310-41 with Xenix Networking Software, iNA961 Transport Software and iSXM 552 Transport Engine
SYS 310-17XN	XENIX System 286/310-17 with Xenix Networking Software, iNA 961 Transport Software and iSXM 552 Transport Engine
iMDX 457	Ethernet Transceiver Cable
iMDX 3015	Ethernet Transceiver
iMDX 3016-1	Ethernet Cable
iDCM 911-1	Intellink™

Ethernet hardware and software for the IBM Personal Computer is available from Ungermann Bass, Inc.

OpenNET™ Personal Computer Link

- **Connects an IBM* PC AT, PC XT (and PC-DOS Compatibles) to the OpenNET Network**
- **Works With Standard DOS Commands**
- **Interconnects a PC System to iRMX™, XENIX*, and NDS-II/NRM Systems Offering OpenNET Server Capability**
- **Uses an 80186/82586 Processor-based Network Controller Board**
- **Supports the ISO/OSI Seven Layer Networking Standards**
- **Enables a PC System to Access Remote Storage and Printer Devices**
- **Provides Transparent-file-access Capability Between a PC System and Remote Servers**
- **Contains Power-up Diagnostics**
- **Uses ISO 8073 Transport and Ethernet/IEEE 802.3 Standard Communication Protocols**

The OpenNET Personal Computer Link (OpenNET PC Link) enables users to connect their IBM PC AT and PC XT computer systems to the OpenNET network. This connection enables a PC system to be configured as a consumer workstation on the OpenNET network, and to transparently access and share files and printers on an OpenNET network resource manager (NRM), NDS-II (with the OpenNET upgrade installed) NRM, iRMX, and XENIX-based remote server systems. The OpenNET PC Link is an 80186/82586 microprocessor-based expansion board, which is easily installed in an expansion card slot of the PC system. On-board jumpers and a user configurable software package enable the OpenNET PC Link to be used with a wide-range of expansion boards currently available for the PC system. The OpenNET PC Link incorporates the Microsoft* Networks (MS-NET) networking software and iNA 960 (ISO 8073 compatible) transport software as a part of its software package.

*IBM is a registered trademark of the International Business Machines Corporation.

*XENIX is a trademark of the Microsoft Corporation.

*Microsoft is a registered trademark of the Microsoft Corporation.

OCTOBER 1985
Order Number: 280164-001

PRODUCT OVERVIEW

The OpenNET PC Link is a member of Intel's OpenNET networking product family. The OpenNET products incorporate a set of system and component level LAN products covering all seven layers of the ISO (International Standards Organization) Open System Interconnect (OSI) model, and the protocols on which they are based. OpenNET network protocols are established industry standards for each function. Therefore, OpenNET network products can connect and operate not only with each other, but with the most popular networking products from other vendors. OpenNET networks provide a high level of interoperability between heterogeneous systems (MS-DOS*, PC-DOS, iNDX, XENIX, and iRMX operating system versions are available). Thus, users can tailor their networks to meet their specific needs by incorporating any combination of the capabilities of these diverse systems.

The OpenNET network application protocols implemented by OpenNET PC Link software are those adopted by Intel, Microsoft, and IBM for their computer networking products. The OpenNET PC Link software is compatible with and will operate with iNDX, XENIX, and iRMX networking software at the application layer.

PHYSICAL DESCRIPTION

The OpenNET PC Link consists of a network controller board and a 5-1/4 inch disk that contains the software necessary for the PC system to communicate across the OpenNET network. The following sections describe the hardware and software components of the OpenNET PC Link.

OpenNET™ PC Link Network Controller Board

The network controller board is an adaptor board that can be installed in any available expansion slot of a PC system. The board implements the industry standard ISO 8073 transport protocol (a modified version of iNA 960) and Ethernet/IEEE 802.3 physical data link technology (see Figure 1). The board uses an Intel 80186 microprocessor in combination with an 82586 LAN communication controller. The board includes the following major components:

- 80186 microprocessor
- 82586 LAN communications controller
- 8 KB of EPROM
- 128 KB of RAM shared between the PC system and the 80186 microprocessor on the network controller board
- 82501 Ethernet serial interface
- Fujitsu MB502A encoder decoder
- 15-pin Ethernet D connector
- 8-bit parallel DMA interface and control register set
- Power-up diagnostics

The network controller board performs all network communication functions for the first two layers of the ISO/OSI model (see Figure 2). Layers three and four reside in the modified iNA 960 transport software. The remaining layers (five through seven) reside in the MS-NET networking software on the PC system.

POWER-UP DIAGNOSTICS

An effective diagnostic function is implemented in firmware on the network controller board. This function is invoked at system initialization during both power-up and system reset time. The following list summarizes the functions tested:

- 80186 and 82586 microprocessors
- I/O ports
- Shared memory window
- Interrupt channels
- DMA channel
- Ethernet connection

An on-board LED indicates whether the network controller board failed any of the various test functions.

OpenNET™ PC Link Software

The software is supplied on a 5-1/4 inch double-density disk (360 KB). The following files are included as part of the OpenNET PC Link:

- A specially configured version of iNA 960 transport layer software, called UBCODE.MEM, which operates on the network controller board.
- A DOS interface driver, called XPORT.EXE, which enables DOS programs to access the network controller board.
- The Microsoft Networks (MS-NET) networking software, release 1.0, which enables users to connect with and access remote file servers on the OpenNET network system.

* MS-DOS is a trademark of the Microsoft Corporation.

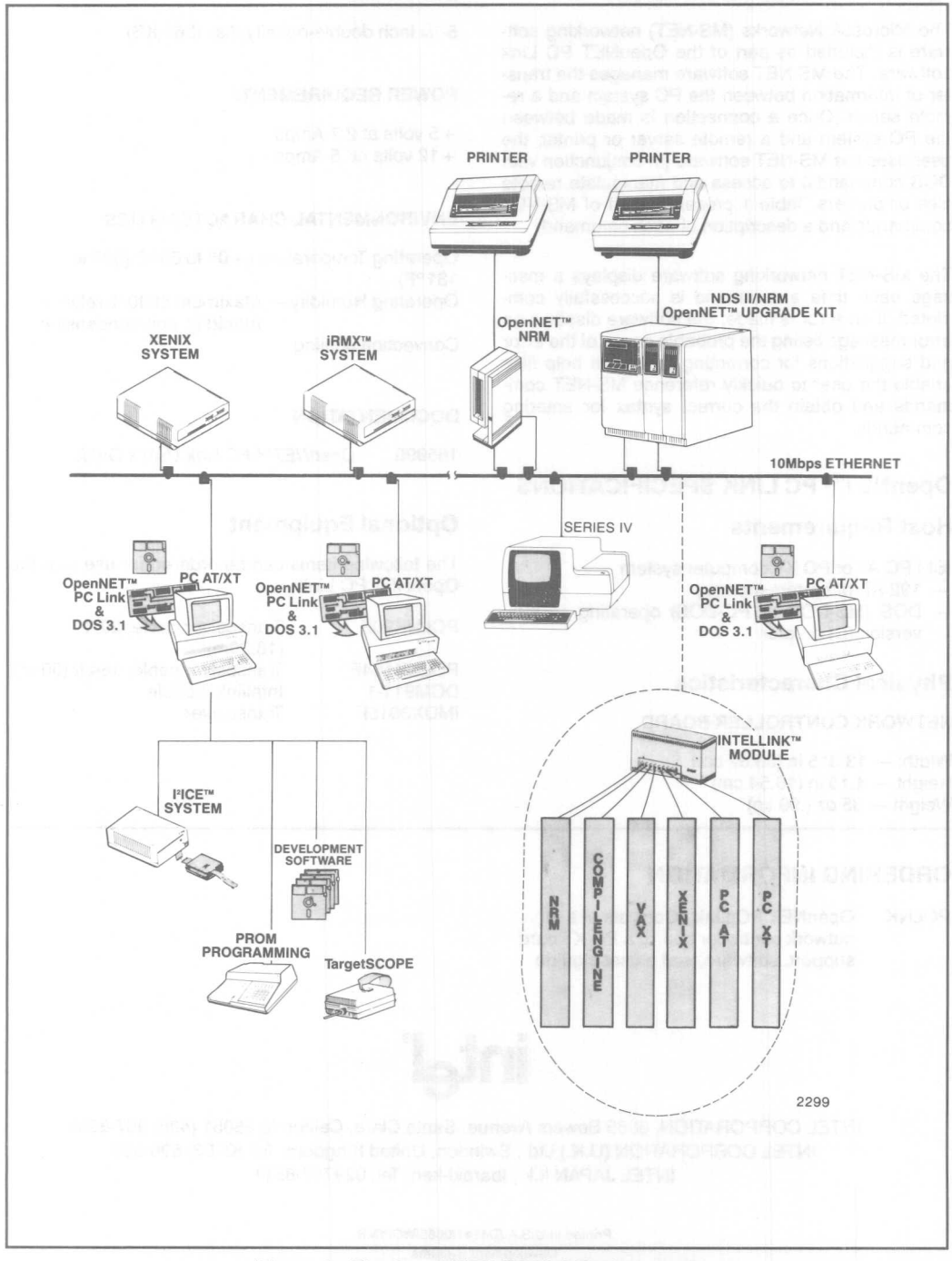


Figure 1. The OpenNET™ PC Link Environment

Microsoft Networks Software

The Microsoft Networks (MS-NET) networking software is included as part of the OpenNET PC Link software. The MS-NET software manages the transfer of information between the PC system and a remote server. Once a connection is made between the PC system and a remote server or printer, the user uses the MS-NET software (in conjunction with DOS commands) to access and manipulate remote files or printers. Table 1 presents a list of MS-NET commands and a description of each command.

The MS-NET networking software displays a message each time a command is successfully completed. If an error is made, the software displays an error message listing the probable cause of the error and suggestions for correcting it. On-line help files enable the user to quickly reference MS-NET commands and obtain the correct syntax for entering commands.

OpenNET™ PC LINK SPECIFICATIONS

Host Requirements

IBM PC AT or PC XT computer system
— 192 KB of system memory
— DOS (MS-DOS or PC-DOS) operating system, version 3.1 or later

Physical Characteristics

NETWORK CONTROLLER BOARD

Width: — 13.315 in (33.82 cm)
Height — 4.15 in (10.54 cm)
Weight — 35 oz (.99 kg)

SOFTWARE

5-1/4 inch double-density disk (360 KB)

POWER REQUIREMENTS

+5 volts at 2.7 Amps
+12 volts at .5 Amps

ENVIRONMENTAL CHARACTERISTICS

Operating Temperature — 0° to 55°C (32° to 131°F)
Operating Humidity — Maximum of 90% relative humidity, non-condensing
Convection cooling

DOCUMENTATION

165996 *OpenNET™ PC Link User's Guide*

Optional Equipment

The following items can be ordered for use with the OpenNET PC Link:

PCLNK20F	Transceiver cable, 20 ft (18.29m)
PCLNK164F	Transceiver cable, 164 ft (50 m)
DCM911-1	Intellink module
iMDX3015F	Transceiver

ORDERING INFORMATION

PCLNK	OpenNET PC Link: Consists of a network controller board, a PC XT card support, software, and a user's guide

intel

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) 987-8080
INTEL CORPORATION (U.K.) Ltd., Swindon, United Kingdom; Tel. (0793) 696-000
INTEL JAPAN k.k., Ibaraki-ken; Tel. 029747-8511

Printed in U.S.A./D4141/0685/WCP/MB
Development Systems

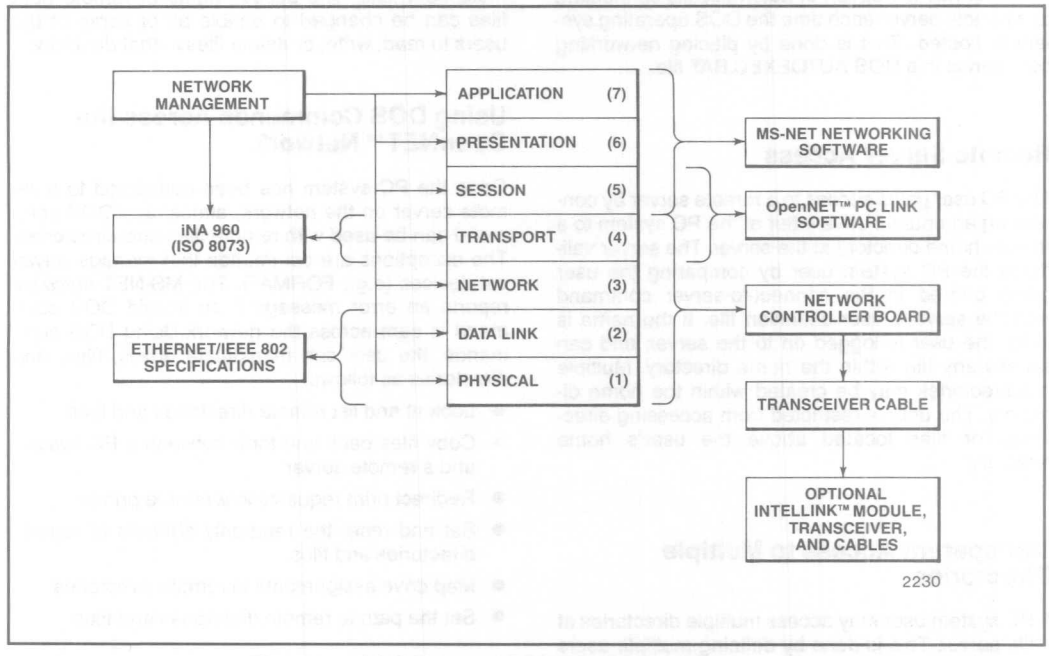


Figure 2. ISO/OSI OpenNET™ PC Link Implementation

PC SYSTEM REQUIREMENTS

For the PC system to function as a workstation on the OpenNET network, it must contain at least 192 KB of memory. A 32 KB memory window is shared between the PC system and the network controller board. The starting address of this window must be placed in an area of memory that does not conflict with the PC system's internal memory address space. The network controller board is jumpered at the factory to reflect a setting which is compatible with the PC system and most of the expansion boards available for use with the PC system.

In order for the network controller board and the OpenNET software to function properly, the PC system must use the DOS (MS-DOS or PC-DOS) operating system, version 3.1 or later.

FUNCTIONAL DESCRIPTION

The OpenNET PC Link enables a PC system to be configured as a consumer workstation in the OpenNET network environment. This enables a PC system to access and share files and remote printers on a remote file server. After establishing a connection with a remote server, the user can access different directories by connecting drive letters at the PC system to the desired directories.

Creating a PC System Consumer Workstation

The PC system is easily configured as a consumer workstation on the OpenNET network. The following steps summarize how to configure the PC system for use as a workstation:

- Install the OpenNET PC Link network controller board in the PC system and connect the PC system to an Ethernet transceiver or Intellink™ module.
- Configure the OpenNET PC Link software to reflect the name and network address assigned to the PC system and each remote server system that the PC system will access.
- Define the PC system user as a valid user of the remote server system.

To connect with and access remote resources over the OpenNET network, perform the following steps:

- Invoke the PC system's consumer networking software.
- Execute a connect-to-server command.
- Execute standard DOS commands.

The PC system user can now access remote resources (files, directories, or printers) at remote servers on the network.

The user has the option of automatically connecting to a remote server each time the DOS operating system is booted. This is done by placing networking commands in a DOS AUTOEXEC.BAT file.

Remote Server Access

The PC user gains access to a remote server by connecting an unused drive letter at the PC system to a remote home directory at the server. The server validates the PC system user by comparing the user name offered in the connect-to-server command with the server's user definition file. If the name is valid, the user is logged on to the server, and can access any file within the home directory. Multiple subdirectories may be created within the home directory. The user is restricted from accessing directories or files located above the user's home directory.

Transparent Access to Multiple Directories

A PC system user may access multiple directories at a file server. This is done by defining multiple users (giving users access to different directories) at the server. After establishing a connection to the server, the user can access different directories by connecting drive devices at the PC system to the desired directories.

Data and resource sharing are implemented via transparent remote file access. This enables the user to work with remote data files and resources residing at server systems on the network as if they were resident on the PC system. Users of a remote server may be given access to the same home directory, enabling multiple users to access and share remote data files. The access rights of remote data files can be changed to enable all or some of the users to read, write, or delete files in that directory.

Using DOS Commands Across the OpenNET™ Network

Once the PC system has been connected to a remote server on the network, almost any DOS command can be used with remote files and directories. The exceptions are commands that manage physical devices (e.g., FORMAT). The MS-NET software reports an error message if an invalid DOS command is sent across the network.Using DOS commands, the user can manipulate drives, files, and directories as follows:

- Look at and list remote directories and files
- Copy files back and forth between a PC system and a remote server
- Redirect print requests to a remote printer
- Set and reset the read-only attribute of remote directories and files
- Map drive assignments to remote directories
- Set the path to remote directories and files

Shared Printer Access

The PC system can be linked to a remote printer that is connected to a server on the OpenNET network. This enables the user to take advantage of the remote printer services, thus freeing the user from having to install a printer at the PC system.

A PC system user can print local or remote data files by first connecting the PC system's logical printer device to the remote server's printer spool. Then, the MS-NET networking software command NET PRINT is used to print the file on the remote printer device.

Table 1. MS-NET Networking Commands

Command	Description
APPEND	Locates a file which is outside the current directory.
NET CONTINUE	Restarts the disk redirector or print redirector programs.
NET HELP	Displays a help file with information about MS-NET commands.
NET NAME	Displays the name assigned to your PC system.
NET PAUSE	Temporarily halts the disk redirector and print redirector programs.
NET PRINT	Prints a file on a remote printer.
NET START REDIRECTOR	Invokes the OpenNET PC Link consumer networking software.
NET USE	Connects a device at the PC system to a remote server or printer.